PET™ GRAPHICS

NICK HAMPSHIRE

A NICK HAMPSHIRE COMMODORE PUBLICATION

PET™ GRAPHICS

NICK HAMPSHIRE

A NICK HAMPSHIRE COMMODORE PUBLICATION

PET™ GRAPHICS

NICK HAMPSHIRE

A NICK HAMPSHIRE COMMODORE PUBLICATION

First Edition April 1981

The programs presented in this book have been included for their instructional value, they have been checked out with care, however, they are not warranted for any purpose. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for any errors or omissions. Neither is any liability assumed for damages or other costs resulting from the use of the information contained herein. No patent liability is assumed for the information contained herein nor do the publishers assume any liability for infringement of patents or other rights of third parties resulting from use of that information. No licence is granted by the equipment manufacturers under any patent or patent rights and manufacturers reserve the right to change circuitry and software at any time without notice. Readers are refered to current manufacturers data for exact specifications.

COPYRIGHT 1981 Nick Hampshire. World rights reserved. No part of this publication may be copied, transmitted or stored in a retrieval system or reproduced in any way including but not limited to; photocopy, photography, magnetic or other recording, without prior written permission from the publishers, with the exception of material entered and executed on a computer system for the readers own use.

Published by Commodore Business Machines (UK) Limited,
818 Leigh Road, Trading Estate, Slough, Berkshire.
Telephone: Slough (0753) 74111 Telex: 848403

PET is a trademark of Commodore Ltd.

Foreword

This book has two objectives. One, to provide the reader with an introduction to the programming techniques used to generate graphic displays. Two, to provide the programmer with a complete package of machine code routines giving a wide range of normally unavailable graphics functions. The reader can key in any of the programs, or enter the machine code using the monitor, but for those unwilling to do this, all the programs in this book are available on disk from Computabits Ltd., P.O. Box 13, Yeovil, Somerset, England, price £10.00, specify size and type of machine. All the programs and routines in this book are written for a 40 column PET with 32K of user memory. Conversion to other size machines is not difficult, study of the comments in the source code will reveal values which must be changed (principaly the column width). All programs have been thoroughly tested over many months and are reasonably free of errors. I would like to thank Mark Witkowski of Queen Mary College London for contributing the work on the PET light pen.

Nick Hampshire.

Contents

Section 1 The Basics of PET Graphics - 1

Section 2 Simple Machine Code Graphics Display Routines - 45

Section 3 Fine Resolution Plotting - 65

Section 4 Screen and Character Block Scrolling - 93

Section 5 Double Density Plotting - 109

Section 6 Multiple Screen Page Displays - 139

Section 7 Displaying and Moving 'Macro characters' - 163

Section 8 A Light Pen for the PET - 177

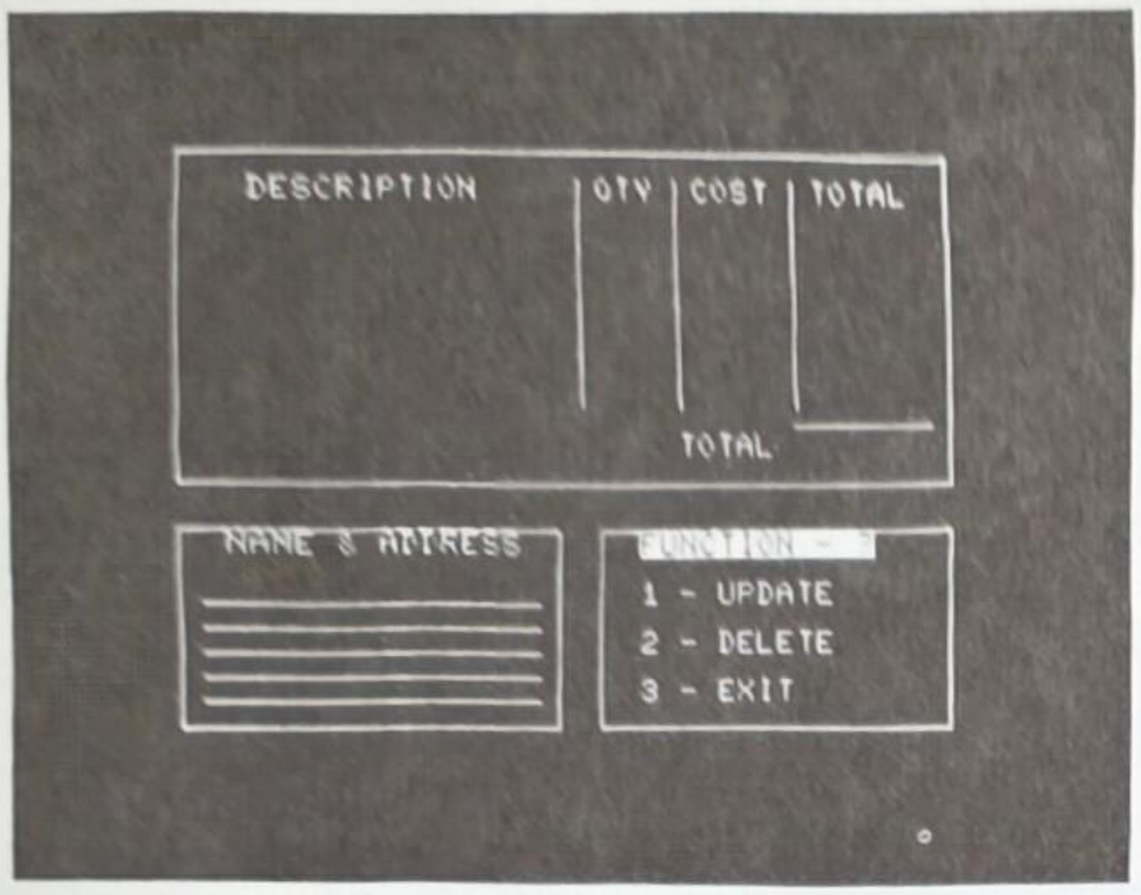
Appendix 1 Circuit Diagrams of PET Video Circuitry - 207

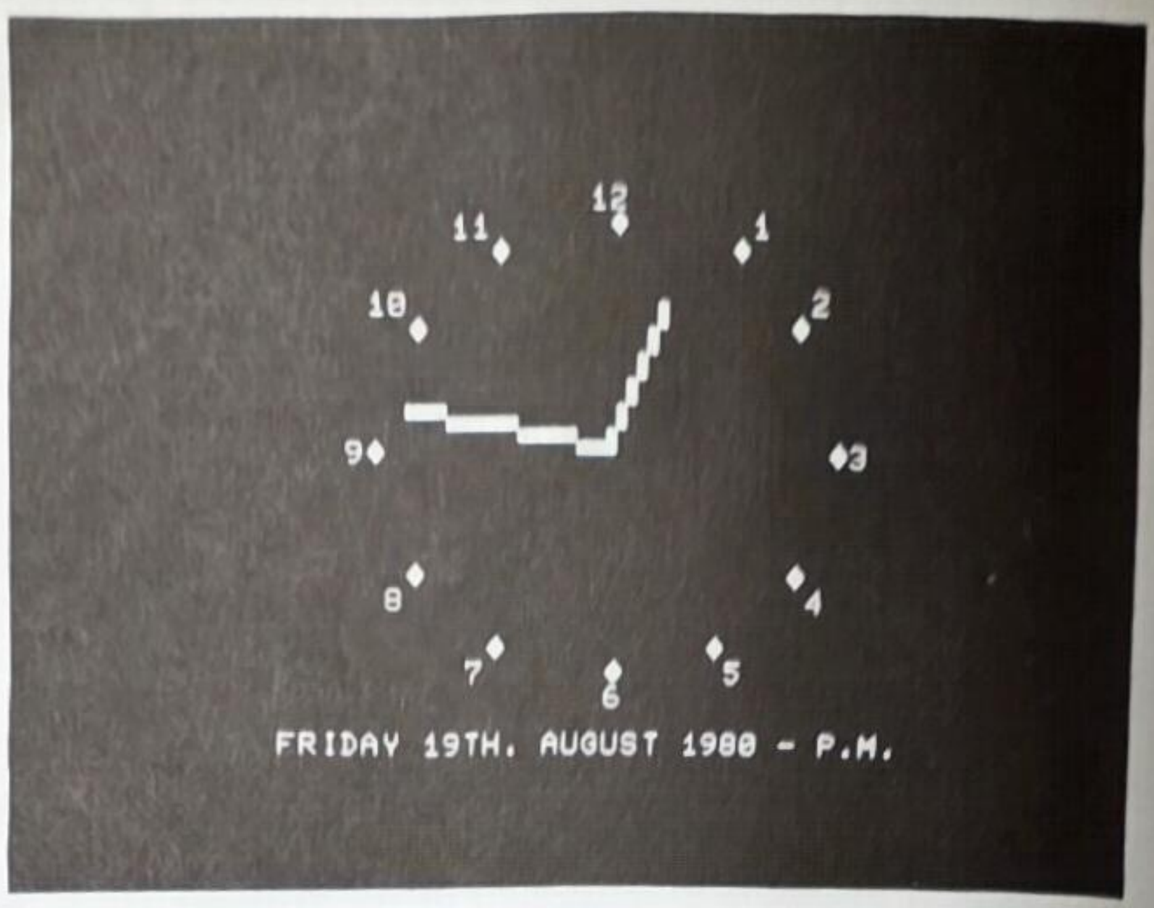
Appendix 2 Table of ASCII codes used by the PET - 210

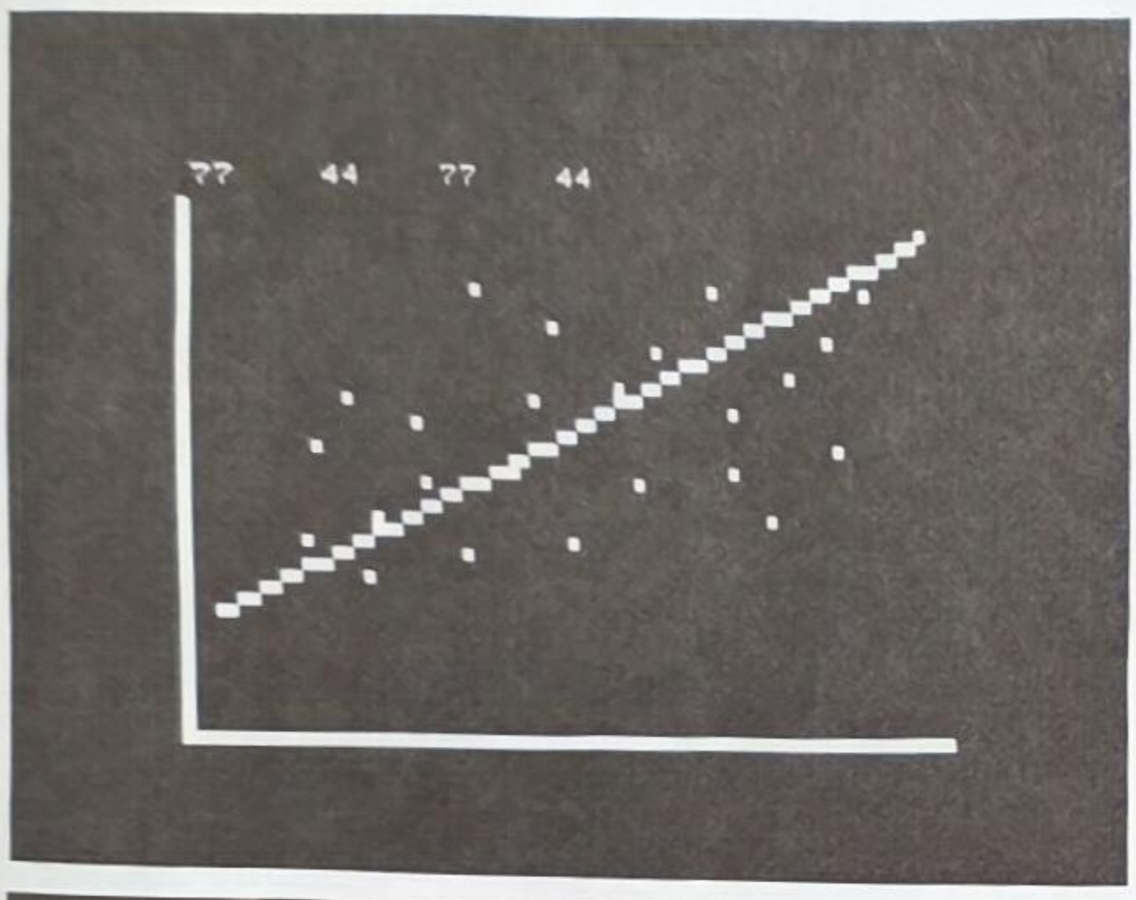
THE BASICS OF PET GRAPHICS

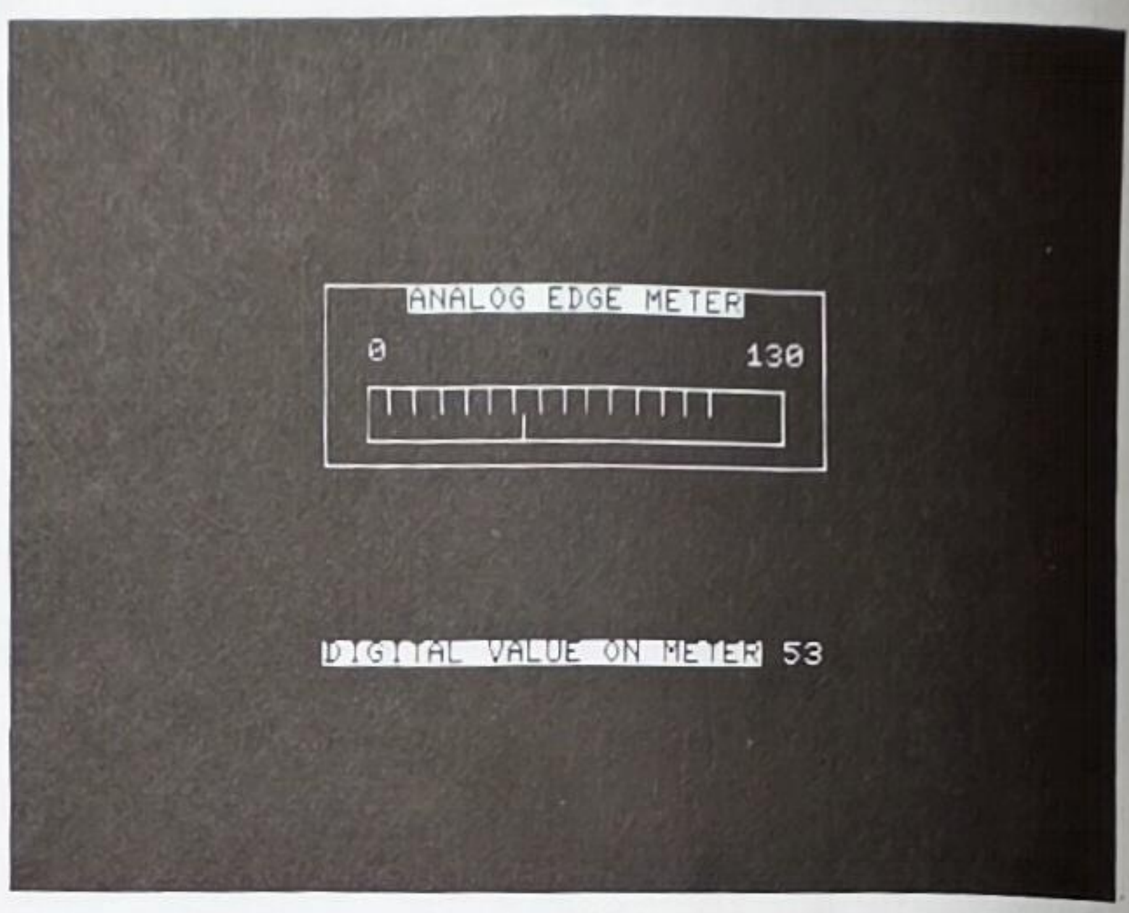
Section 1

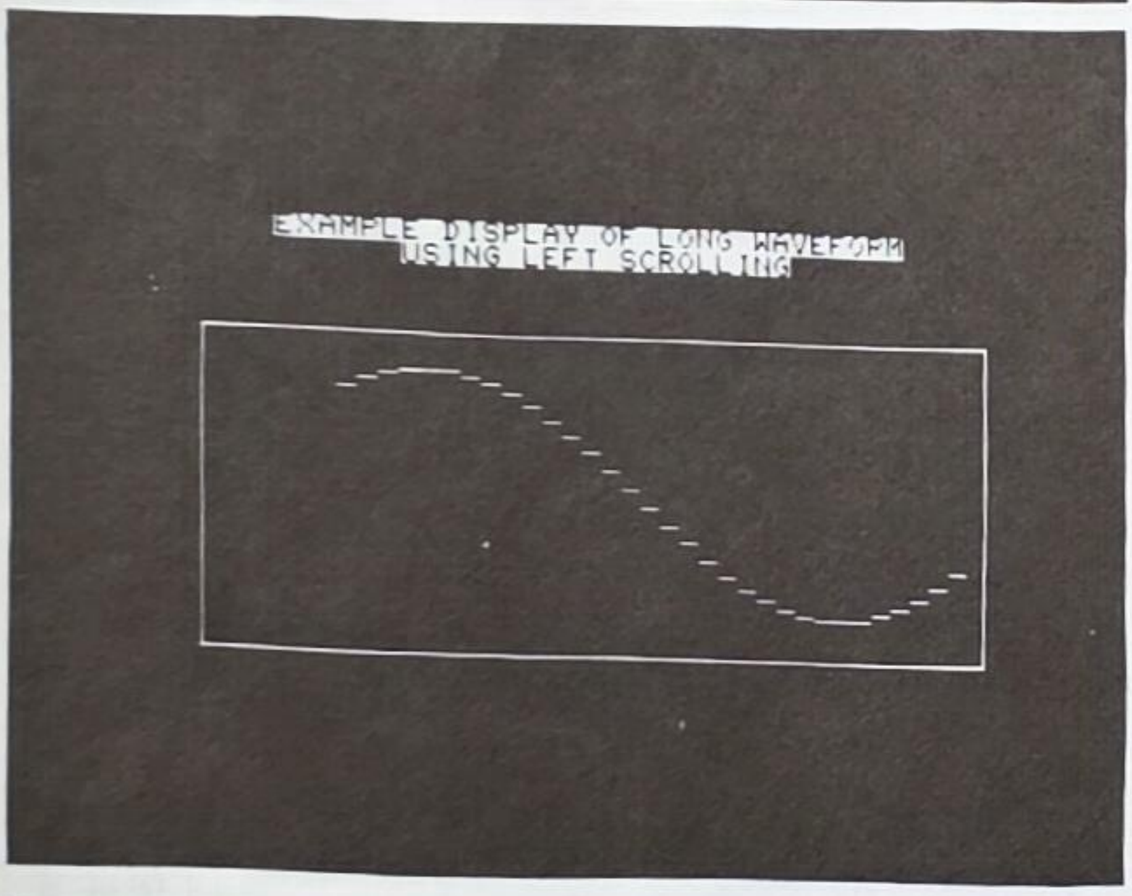
Principles behind operation of video circuitry - 1
Notes on designing a display - 9
The PRINT command and it's applications - 13
Cursor control - 16
'Minefield' a game demonstrating some uses of PRINT - 19
'Hangman' a game demonstrating further uses of PRINT- 22
The POKE command and its application - 26
Using POKE to draw borders, lines and reverse field - 28
Moving characters across the screen with POKE - 30
'Carrace' a game demonstrating some uses of POKE - 32
Graphics displays using machine code subroutines - 34
Introduction to the 'Graphics' package - 34
Assembler format - 37
Variables used in 'Graphics' machine code package - 39
Routine to calculate screen addresse - 40
List of labels used in 'Graphics' package - 42

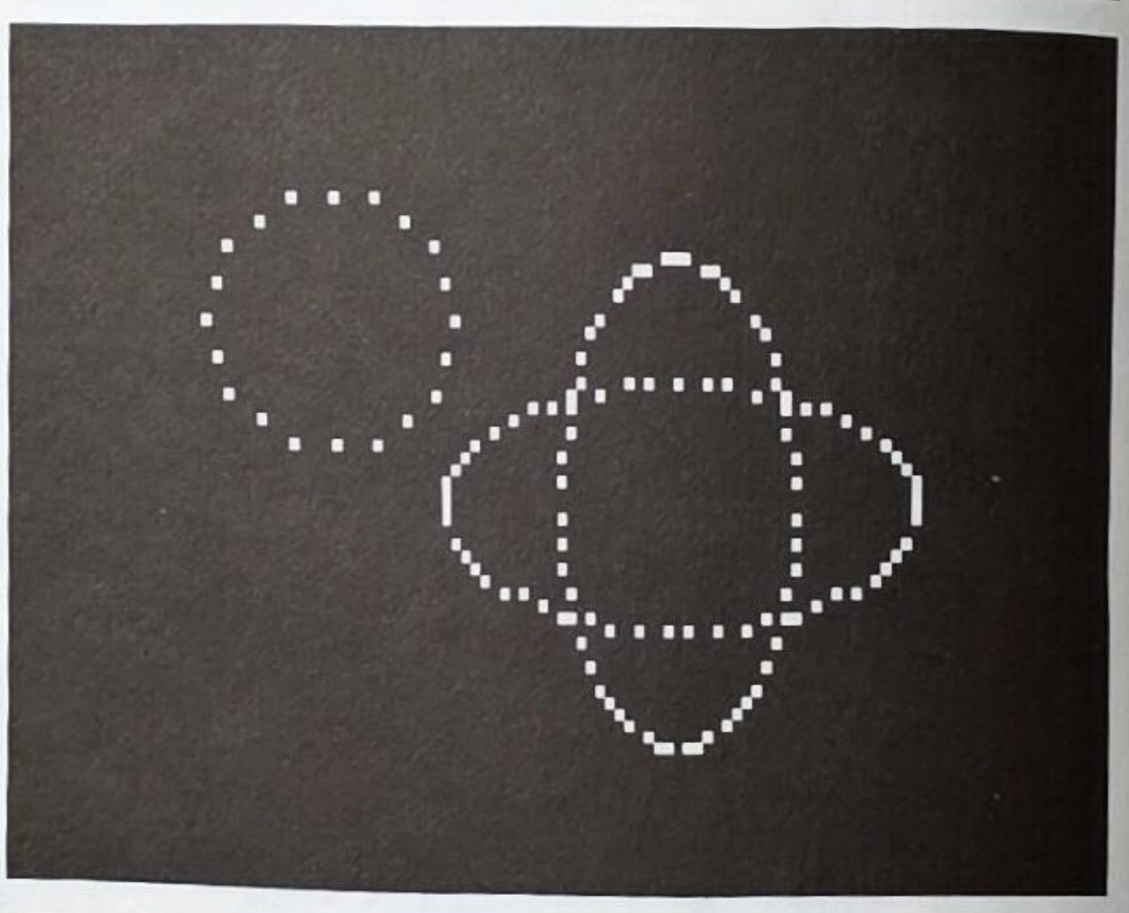

DESCRIPTION
QTY
COST
TOTAL
TOTAL
NAME & ADDRESS
1 - UPDATE
2 - DELETE
3 - EXIT

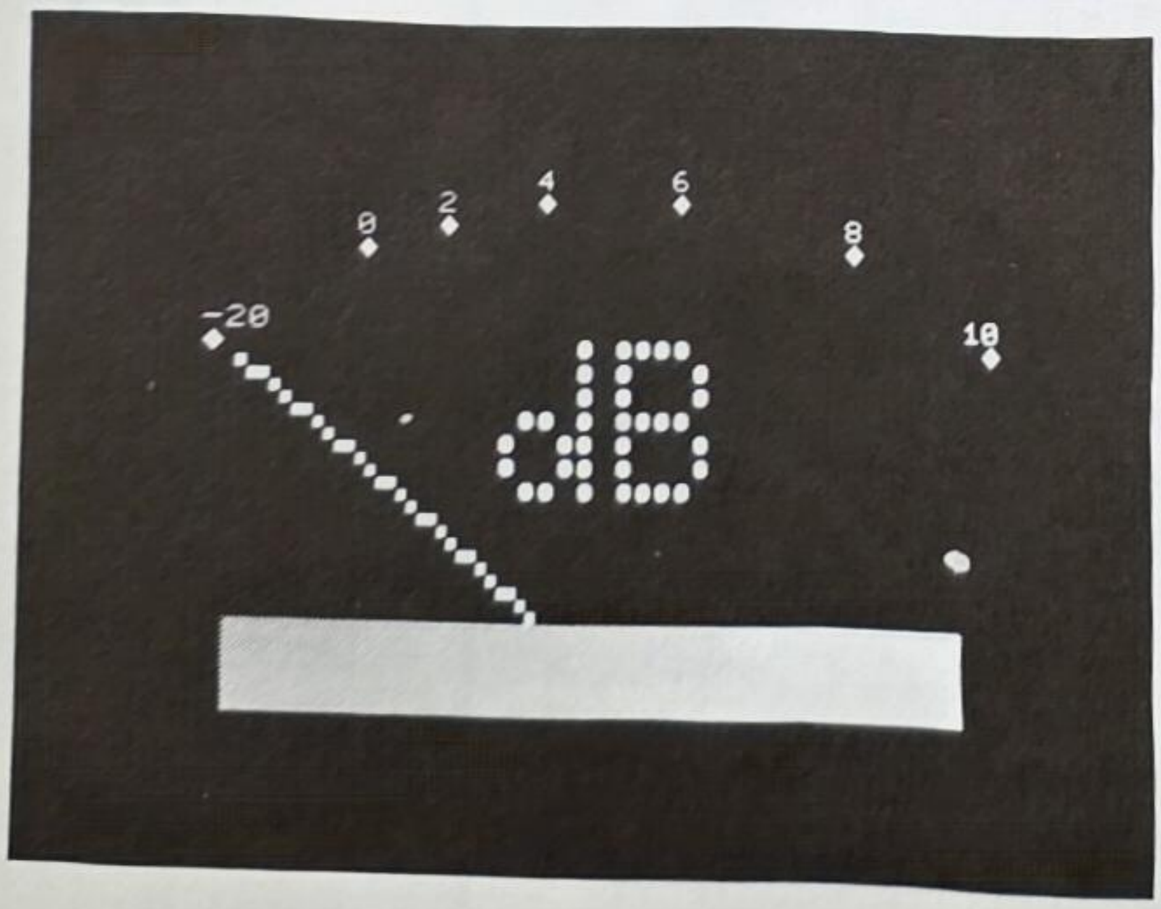

12
1
2
3
4
5
6
7
8
9
10
11
FRIDAY 19TH. AUGUST 1980 - P.M.

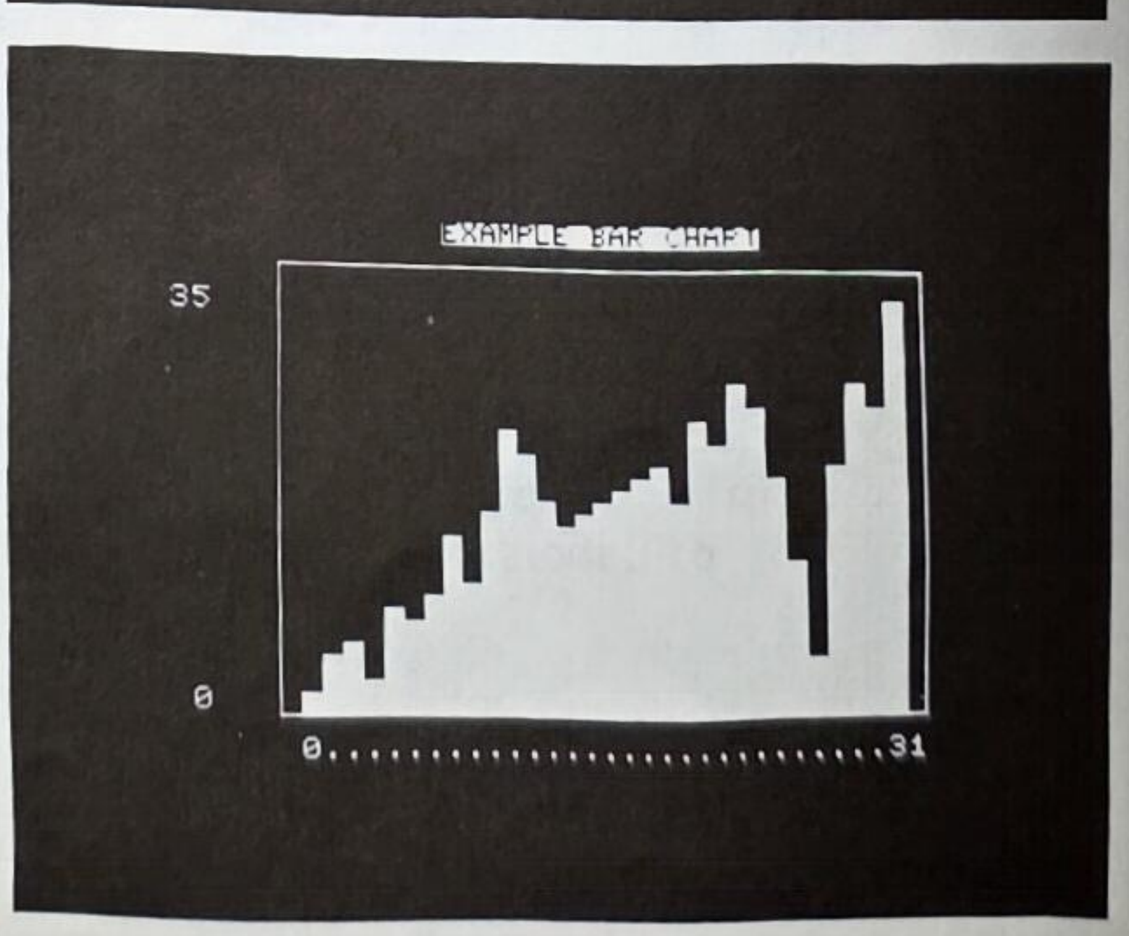

77
44
77
44


ANALOG EDGE METER
0
130
DIGITAL VALUE ON METER 53


EXAMPLE DISPLAY OF LONG WAVEFORM
USING LEFT SCROLLING




0
2
4
6
8
10
-20
dB


EXAMPLE BAR CHART
35
0
0
31

THE BASICS OF PET GRAPHICS 1

Effective use of the graphics capability of the PET requires an understanding of the way characters are generated, stored and displayed. The PET display is constructed around a raster scan TV monitor. In raster scanning the electron beam in the TV tube is deflected back and forth across and down the screen many times a second. The method used in a domestic TV set. To cover the whole screen the electron beam is scanned back and forth 512 times. Fig 1. shows the raster scan lines going across and down the screen. The raster scan does not normaly illuminate the screen although if you turn the screen intensity knob at the back of the display to maximum the scanning lines will usualy become visable. The bright dots on the screen, which make up each character, are produced only when the intensity of the electron beam of the raster scan is increased to a level where the screen phosphor is excited into luminesence. In normal TV broadcasting the TV set receives a signal containing special synchronising pulses. These synchronising pulses are used to ensure the accurate timing and positioning of each raster scan line. There are horizontal sync pulses every 64us for the horizontal sweep of the beam. Vertical sync pulses are every 1/60 second and initiate the vertical trace of the beam. Between the sync pulses is the video information, in the form of pulses, for a single horizontal raster scan line. The three different pulse sequences which comprise the video output to the monitor are shown in Fig 2. These information pulses turn the electron beam on or off, a dot is displayed on the screen's phosphor when the beam is "on".

The 512 horizontal lines are displayed in pairs, a process known as interlacing. The vertical definition of the screen is thus set at a maximum of 256 lines. The amount of information on each horizontal line (the number of pulses or dots on each screen line) depends on the bandwidth of the display circuitry. A normal monitor, such as that on a 3032 PET, has a bandwidth of about 6MHz. The maximum number of dots per line which can be displayed without blurring the dots is 64us x 6MHz = 384 dots. A 40 column PET requires a screen definition of 320 dots per line, to acheve the

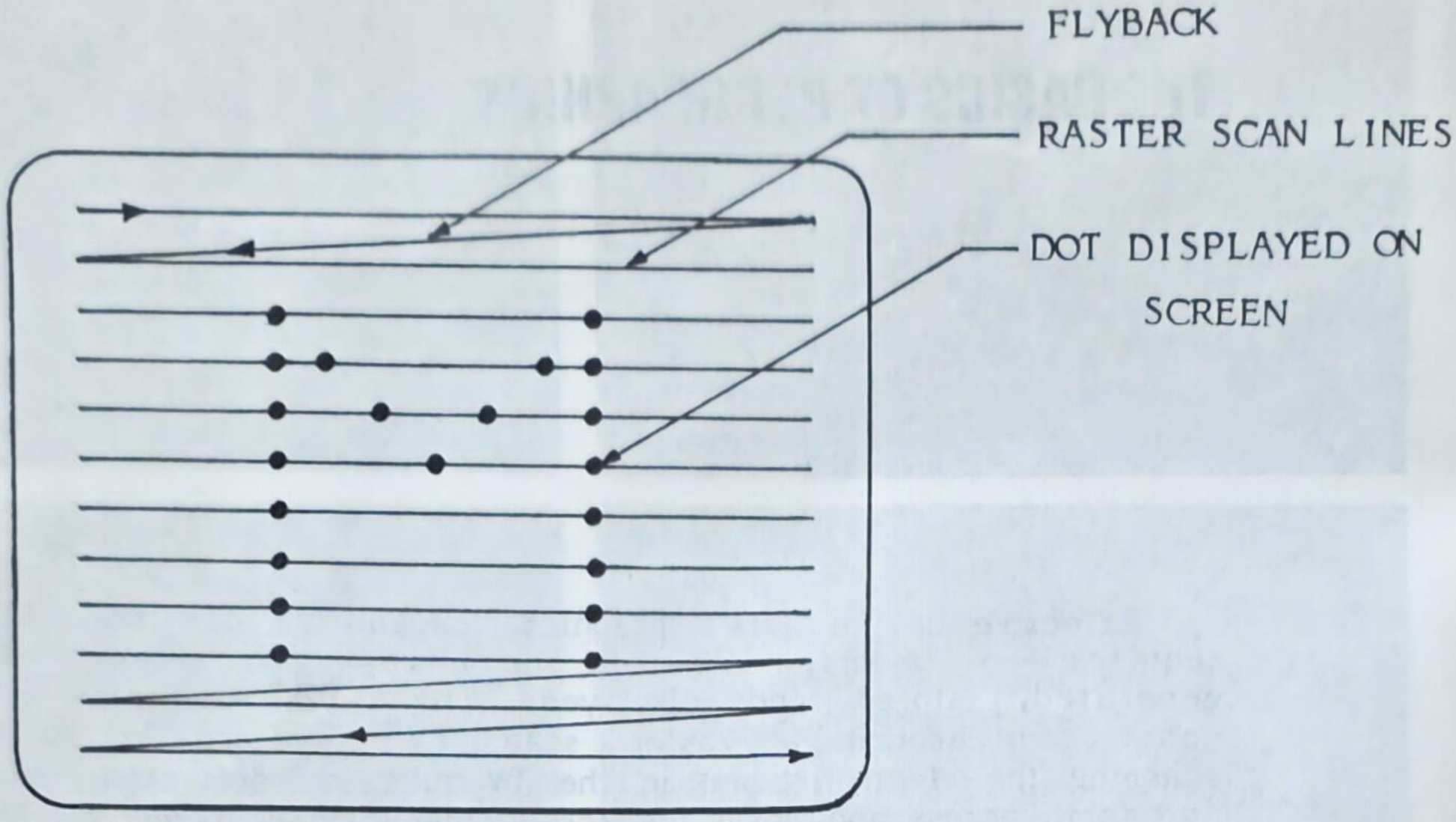
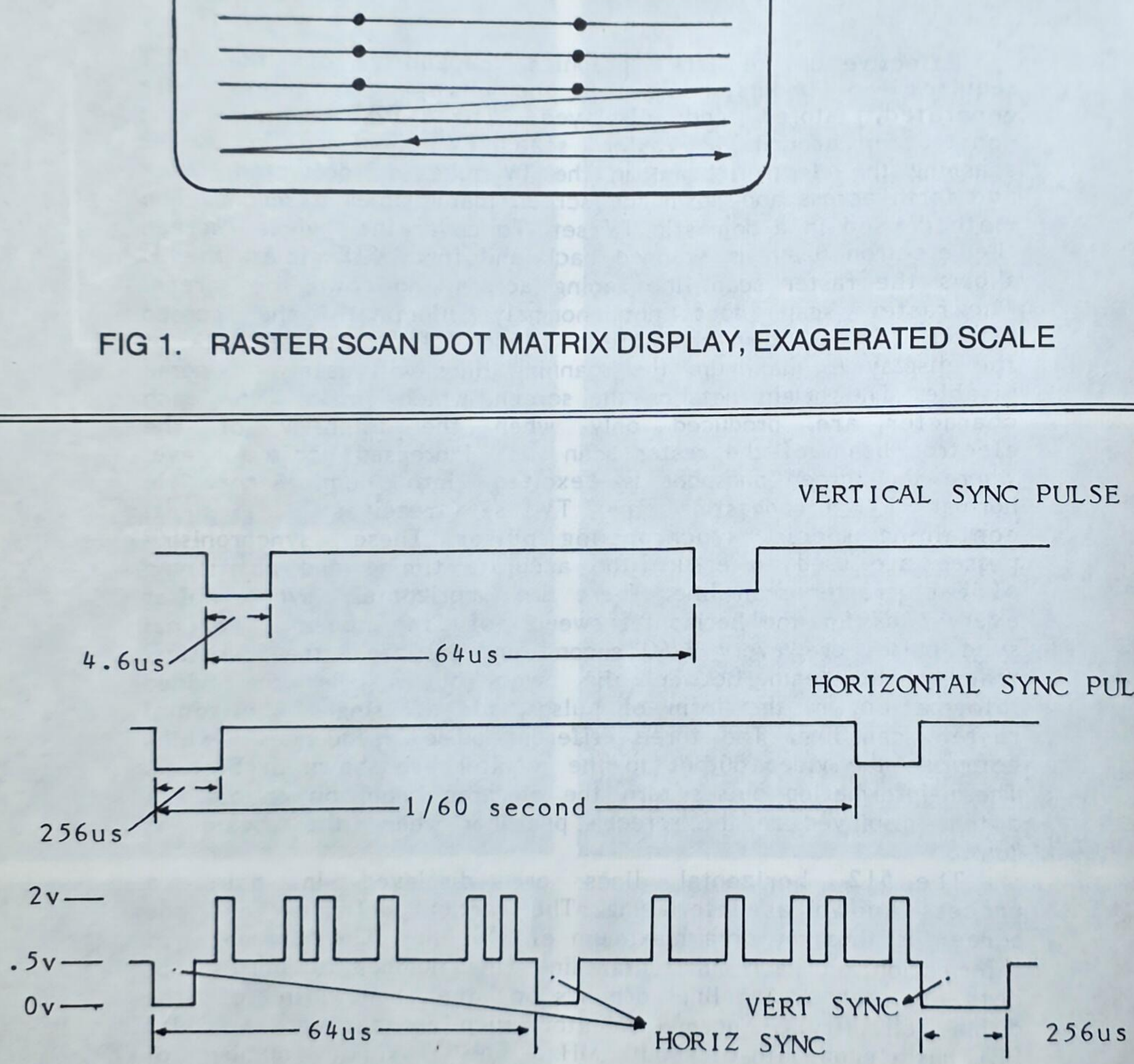


FIG 1. RASTER SCAN DOT MATRIX DISPLAY, EXAGERATED SCALE



Composite video signal from combination of horizontal, vertical and video pulses.

FIG 2. PULSE TIMING DIAGRAM OF VIDEO SIGNALS

definition required for an 80 column machine the video circuitry has been improved to give a bandwidth of about 10MHz. A raster scanned video screen can thus be considered as a matrix of dots, any one of which can be turned on or off by inputing the correct sequence of pulses on a specified horizontal line. The screen definition on a 40 column PET is 320 dots on a horizontal line by 200 lines verticaly, on an 80 column 640 horizontal by 200 vertical.

Shapes can be displayed on the screen by switching the beam on at specific locations. Each character on the PET screen is formed from a small square of eight by eight dots. Each of these 64 dots is selectivly turned "on" or "off" to produce the desired character, individual dots can be seen on closely examining the screen. The task of constructing characters, outputing the information pulses and the correct sync pulses, is performed by the video logic circuitry. The PET video circuit is based on a system known as memory mapped video. The text or graphics characters to be displayed on the screen are stored in a block of memory, shown in Fig 3, this is shared between the video circuit and the computer. On a 3032 PET this area of memory extends from location decimal 32768 to 33767. The command POKE 32768,42 will place a star in the top left hand corner of the screen. In this memory area characters are stored not as 64 bits, each representing

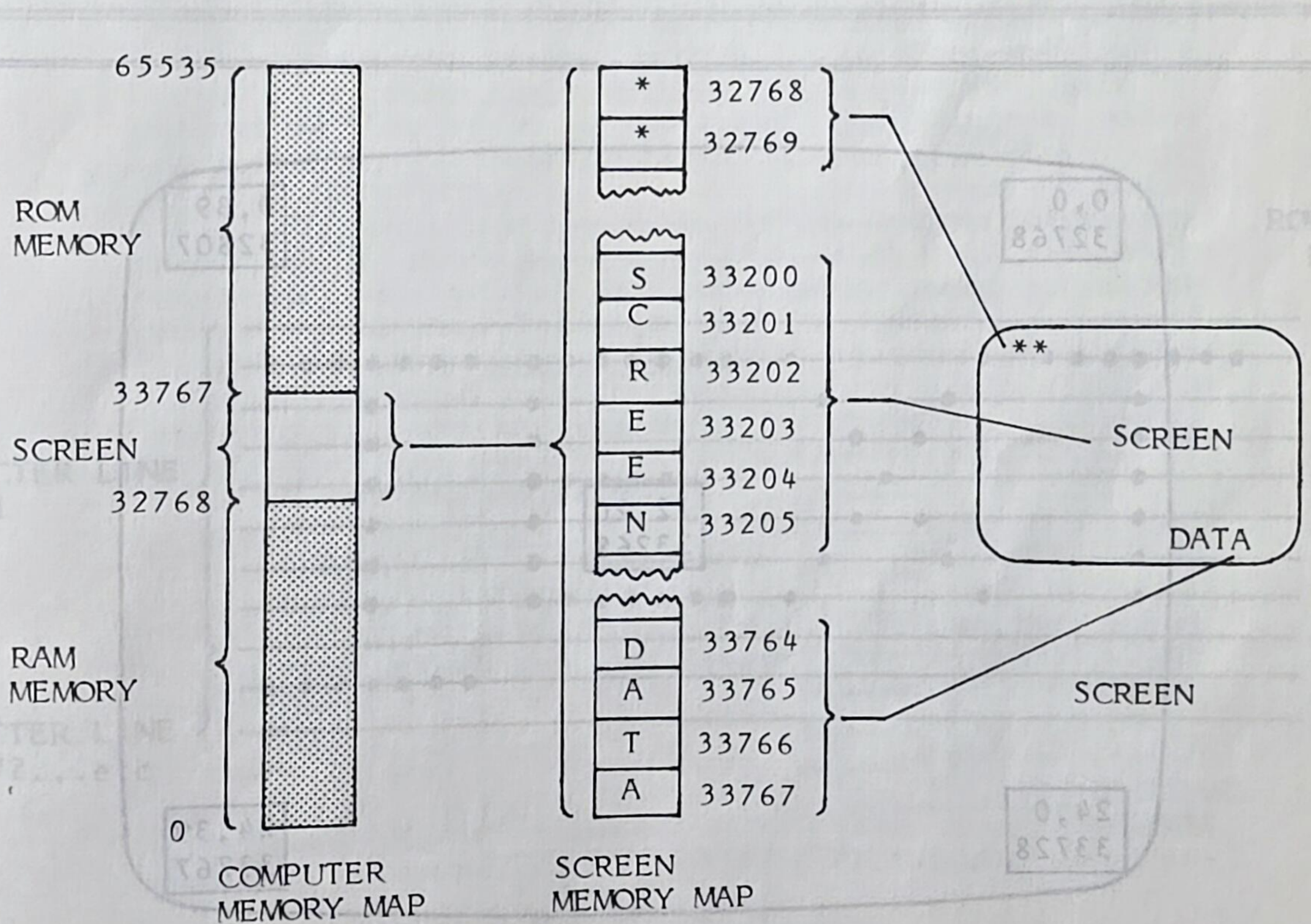


FIG 3. MEMORY MAPPED VIDEO DISPLAY

a screen dot, but as an eight bit ASCII coded byte. Each different character has its own unique ASCII code, the code for a star is decimal 42. The video circuitry converts these codes by means of a character generator into the pattern of dots on the screen. Each location in the screen memory corresponds to a specific location on the screen, where a screen location is a block of 8 x 8 dots. Fig 4, shows the relationship between video memory locations and the screen line and column number. The fact that memory locations directly correspond to screen locations in a memory mapped system gives the programmer a wide range of options in displaying information.

The eight bit ASCII codes stored in the video RAM memory are converted into the pattern of dots displayed on the screen by a device known as a "character generator ROM". The character generator is the central component around which the video circuitry is constructed. The basic functioning of the video circuitry is shown in Fig 5, (the complete circuit is shown in Appendix 1). Each screen memory location is accessed, and the ASCII code contained in that location transfered to the character generator input, by means of the memory scanning circuitry. The character generator converts the code into a stream of serial pulses via a shift register. The whole process being controlled by the timing circuitry which also produces the horizontal and vertical sync pulses.

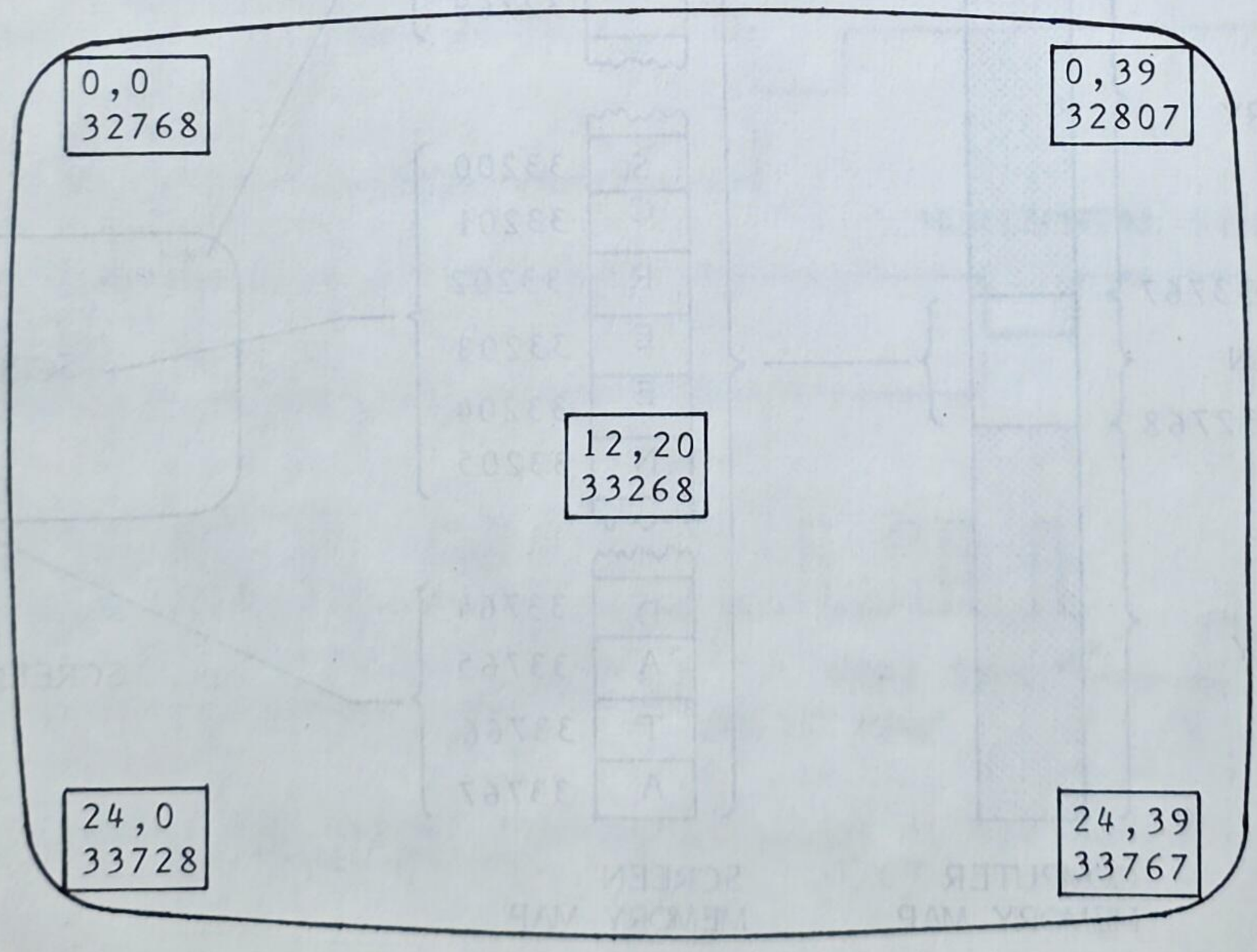


FIG 4. SCREEN MEMORY MAP BY LINE, COLUMN, AND MEMORY LOCATION

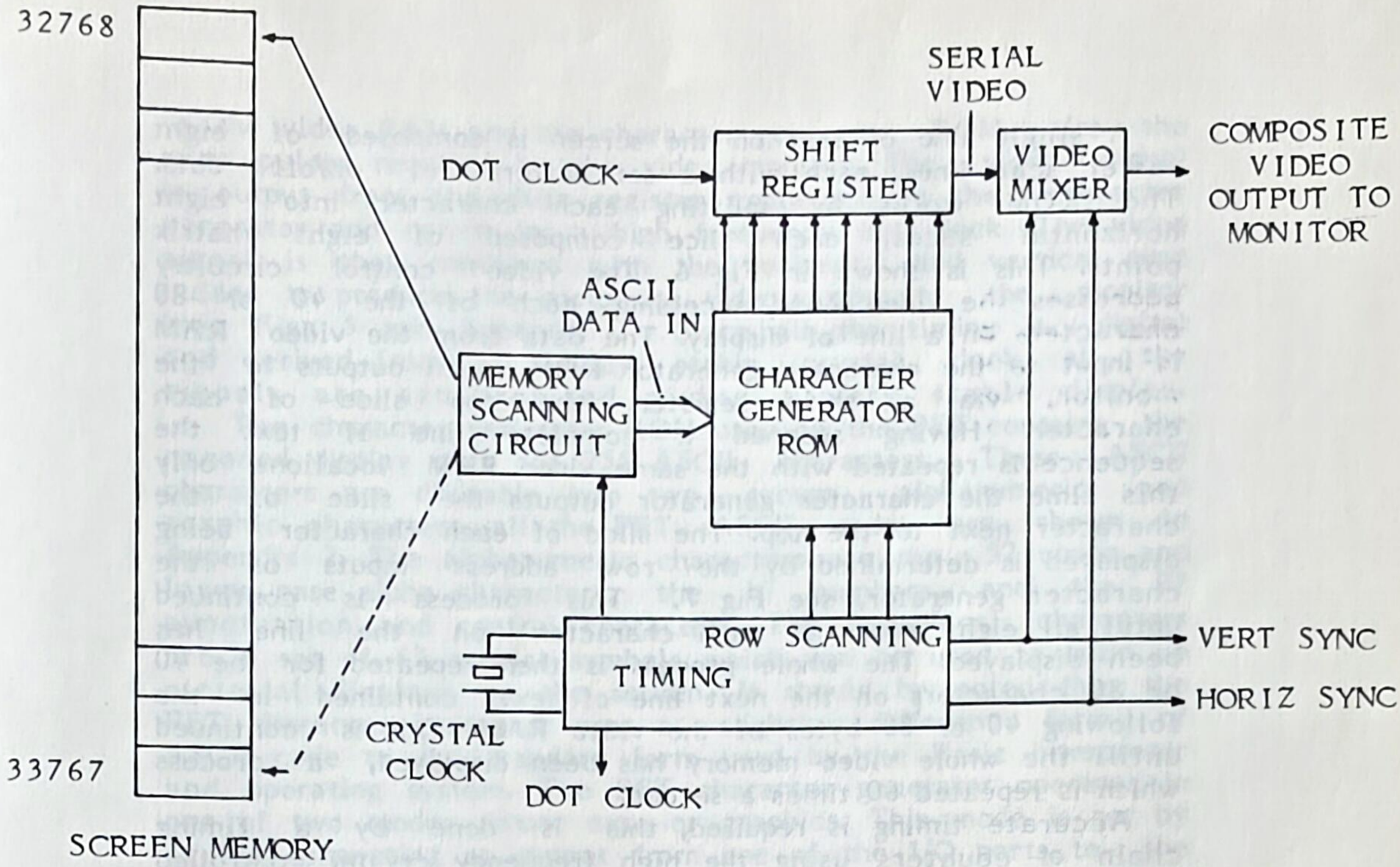


FIG 5. BASIC FUNCTIONS OF VIDEO DISPLAY CIRCUIT

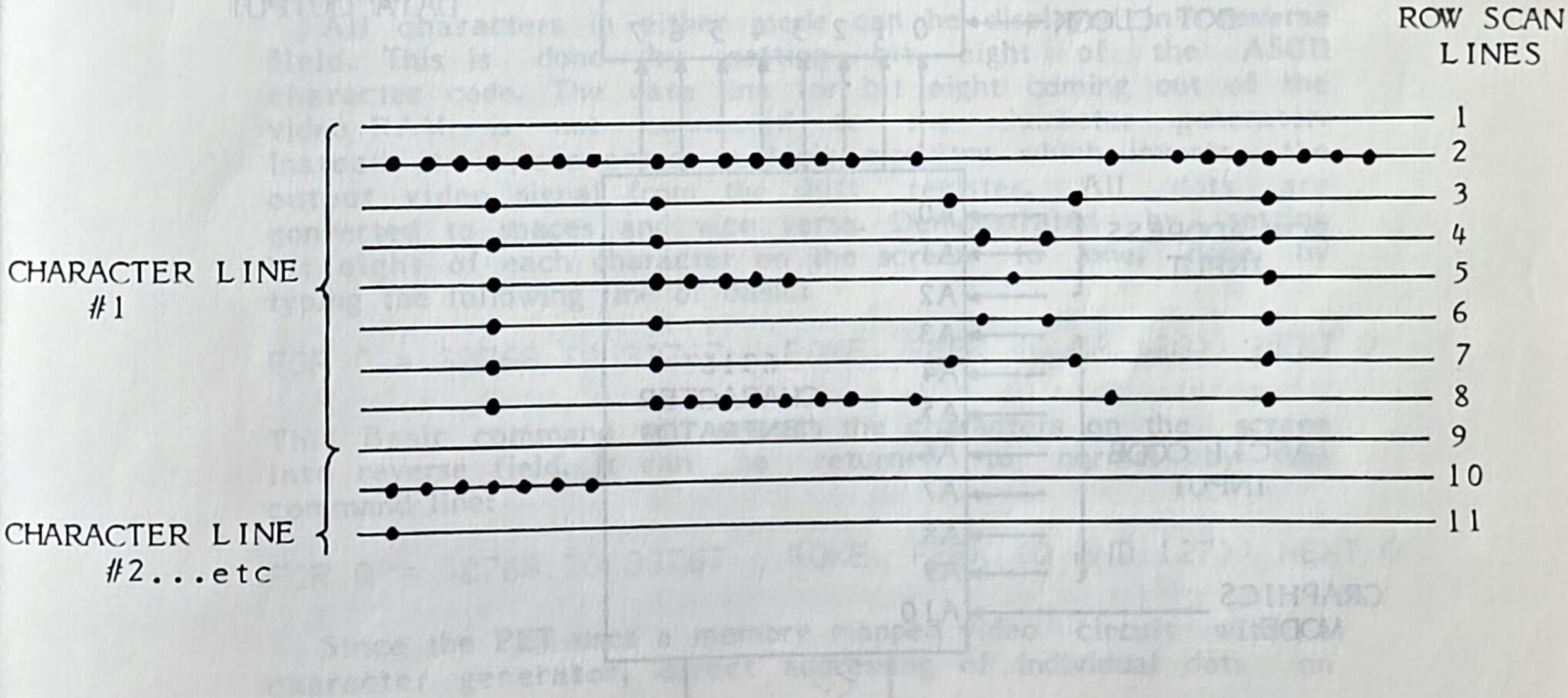


FIG 6. CHARACTERS ARE DISPLAYED ONE LINE OF DOTS AT A TIME

A single line of text on the screen is composed of eight raster scan lines, each with a set pattern of on/off dots. The ROM works by splitting each character into eight horizontal slices, each slice composed of eight matrix points. This is shown in Fig 6. The video control circuitry addresses the video RAM, accessing each of the 40 or 80 characters on a line of display. The data from the video RAM is input to the character generator ROM which outputs to the monitor, via a shift register, the top slice of each character. Having scanned a complete line of text the sequence is repeated with the same video RAM locations only this time the character generator outputs the slice of the character next to the top. The slice of each character being displayed is determined by the row address inputs on the character generator, see Fig 7. This process is continued untill all eight slices of each character on the line has been displayed. The whole process is then repeated for the 40 or 80 characters on the next line of text contained in the following 40 or 80 bytes of the video RAM. This is continued untill the whole video memory has been displayed, a process which is repeated 60 times a second.

Accurate timing is required, this is done by a timing chain of counters using the high frequency crystal controlled system clock. The timing chain generates the address inputs

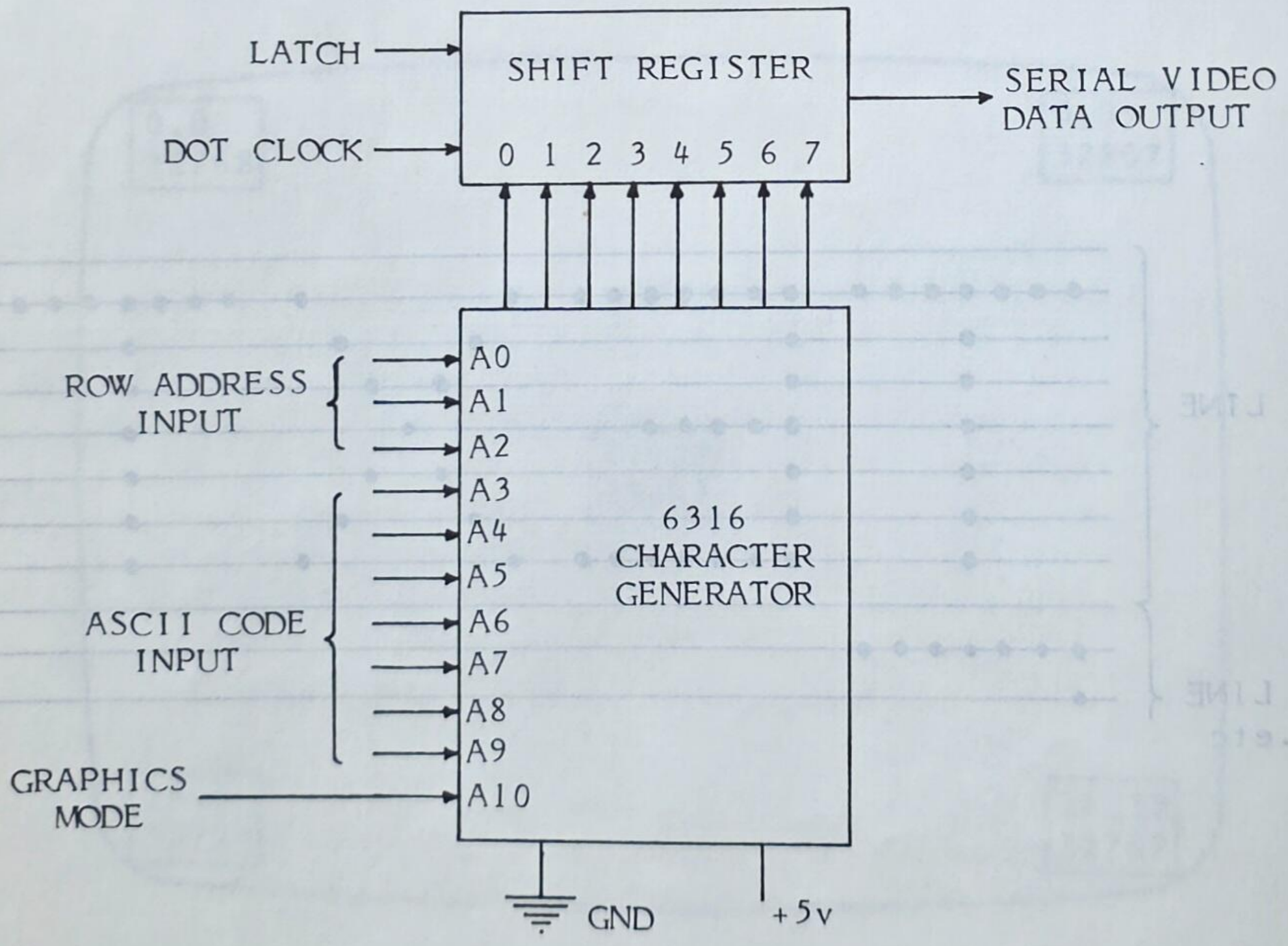


FIG 7. CHARACTER GENERATOR FROM FEEDING A SHIFT REGISTER

to the video RAM and the character generator ROM, also the sync pulses required by the video monitor. The video signal is output from the shift register connected to the character generator and driven by a high frequency dot clock The video signal is then combined with the horizontal and vertical sync pulses to produce the composite video output to the monitor (see Figs 5 and Appendix 1). Since all the timing is digital and derived from a highly stable crystal clock all the signals are synchronised giving a very stable display.

The character generator ROM used on the PET contains the encoded display data for 255 ASCII characters. These ASCII characters are divisable into two groups, alphanumeric and graphic characters, all the PET ASCII codes are shown in Appendix 2. The alphanumeric characters are the 52 upper and lower case alpha characters, the 10 numbers, and the 27 punctuation and control characters. The graphics characters are a set of 63 special symbols which can be used to build up pictorial displays on the screen. It should be noted that the PET character generator uses a slightly different form of ASCII code to the standard form used by the Basic interpreter and operating system. The PET character generator operates in one of two modes, lower case or graphics. This mode is set by a line connecting an output from one of the I/O ports to the most significant address line of the character generator ROM, see Fig 7. This has the function of splitting the character generator into two seperate character sets, which character set is used to create the screen display depends on the state of the most significant address line. Because the character generator ROM is split into two 127 character blocks, many of the characters are repeated in both blocks. When the graphics mode is set, upper case characters and numbers can still be diplayed as they are in the lower case mode, for this reason there are only 152 different ASCII characters out of the potential 255.

All characters in either mode can be displayed in reverse field. This is done by setting bit eight of the ASCII character code. The data line for bit eight coming out of the video RAM is not connected to the character generator. Instead it is connected to logic circuitry which inverts the output video signal from the shift register. All dots are converted to spaces and vice versa. Demonstrated by setting bit eight of each character on the screen to one, done by typing the following line of Basic:

```
FOR Q = 32768 TO 33767  POKE, PEEK (Q OR 128): NEXT Q
```

This Basic command will set all the characters on the screen into reverse field. It can be returned to normal by the command line:

```
FOR Q = 32768 TO 33767 : POKE, PEEK (Q AND 127): NEXT Q
```

Since the PET uses a memory mapped video circuit with a character generator, direct addressing of individual dots on

the 320 x 200 screen matrix is impossible. True high resolution point plot or bit mapped graphics cannot be acheved without additional circuitry, this is dealt with in greater detail later in this book. Point plot graphics means that the screen is divided into a grid and a bright square is displayed at specified coordinates on that grid. The maximum point plot resolution achevable on a 3032 modified for high resolution is a grid with 320 divisions in the horizontal and 200 in the vertical axis. With this resolution each point plot square is composed of a single screen dot (pixel).

By careful use of the 63 graphics characters in both normal and reverse field mode, it is possible to construct graphic displays with the resolution of single dot point plot graphics. This form of graphics display is rather limited and the quality of output is inferior to that obtained from a bit mapped display. In some cases, such as graph plotting, point plot graphics are essential. Unfortunately this means using a single character for each point plot. With this the resolution is limited on a 3032 to 40 divisions on the horizontal and 25 on the vertical. This is very limiting. The resolution of the screen can be improved however, from 40 x 25 on the 3032 to 80 x 50, and on the 8032 from 80 x 25 to 160 x 50, by clever use of the 9 quarter and half square graphics characters. This method means that a point plot dot has dimensions of 4 x 4 screen dots. Double density point ploting as it is known, especialy on the 8032, gives an acceptable resolution for the plotting of graph. Double density graphics are looked at in detail later together with variations of the method which can give even higher resolutions though only in one axis.

Data can be displayed on the screen in one of three ways, these are: using a PRINT statement, with a POKE command and using special routines written in machine code to perform specific display functions. All three of these methods will be looked at in detail in subsequent sections of this book. The easiest to use of the three is the PRINT command, this displays data in either string or simple variable form, on the screen to the right of and including the current cursor position. The PRINT command has the following constraints:

1). The position on the screen at which data is displayed is dependent on the position of the cursor.

2). Data to be displayed by a PRINT command is either in string or numerical variable form. Strings can be up to 80 characters long and numerical variables up to 9 digits.

3). Where the data in either string or numerical form is more than one byte long the first byte is placed under the cursor and all subsequent bytes are displayed sequentialy to the right. Where the edge of the screen is encountered the data is continued at the start of the next line down. Data displayed using PRINT can have multiple bytes which are displayed sequentialy.

The POKE command can be used to display data on the screen because the PET uses a memory mapped video display. POKE will place a value into a single byte of memory, if the value is an ASCII code and the memory is the video memory then a specific character represented by that code will be displayed on the screen at a specific location dependent on the memory address chosen. The POKE command has the following constraints:

1). Only one character can be displayed by a single command step.

2). The address in video memory must be calculated for each character displayed since data displayed using POKE is displayed randomly on the screen (ie. is independent of cursor position).

Specific display functions using machine code routines also make use of the PET's memory mapped video display. Such machine code routines could be written using POKE commands however this in general is far too slow. The great advantage in writing such routines in machine code is not just that they are very fast, a screen can be filled with data almost instantaineously, but that they are under the control of the programmer and can thus be tailored to an individual requirement. Unfortunately although the most useful method it is also the most difficult requiring an understanding of machine code programming. To help overcome this dificulty 'PET Graphics' has been written around a collection of such routines, all of which are fairly general purpose, and can be used without any particular knowledge of machine code.

Whatever method is used to display data on the screen considerable thought must be given to its positioning and the overall apperance of the display. Without such thought to positioning and apperance, a display can look very muddled and consequently difficult for the operator to locate the required information. The layout of a set of displays is an important preliminary task in the design and writing of any program. Any program from games to business applications consist of a controlled sequence of input and output stages. At each of these stages one or more items of data may be input or displayed on the screen, and whether the data is being input or output it is displayed in an orderly form very often accompanied by written explanation. For each input/output stage there is a corresponding screen format which is designed when the program is written. A program may require just one or two different screen formats which are repeatedly used, or a dozen some of which may be used only once.

Each individual screen format is designed in two stages. The first stage is to define what information is to be input or output and what explanations or graphic diagrams,lines, borders etc,accompany them. The second stage is to lay out

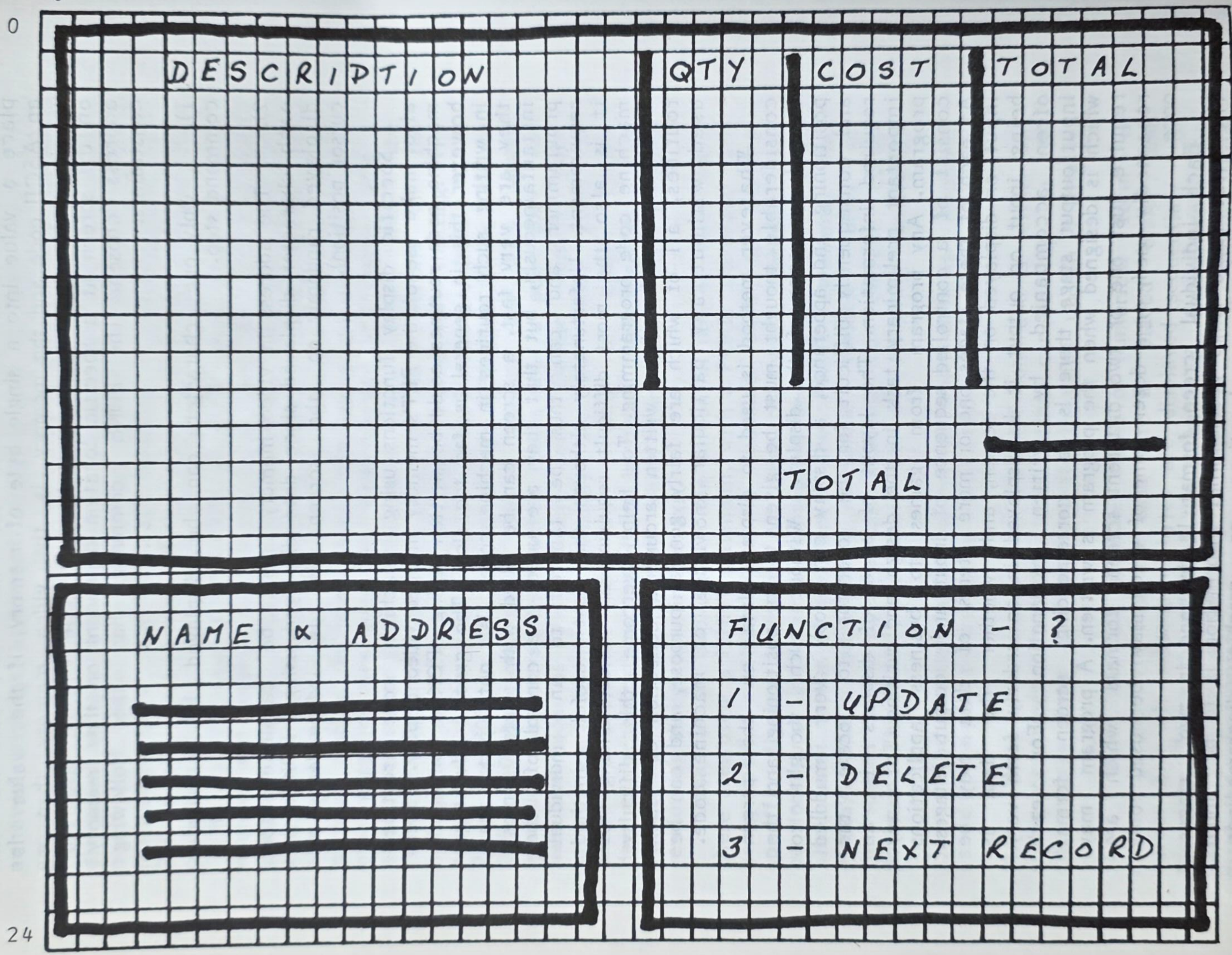


FIG. 8 SAMPLE DESIGN OF SCREEN LAYOUT

the actual apperance of the screen on a sheet of squared paper with the dimensions of the screen (Fig 8 is a sample design). There are no hard and fast rules over how the first stage of screen format design should be done, except that the amount of information displayed on the screen should be kept to a minimum. By keeping the information displayed to a minimum one makes it easier for the operator to assimulate the information, this can be assisted by using the graphics capability to highlight items and associate related information. Information can be highlighted by using reverse field characters, by underlining or by enclosing within a border or a special character field. Associated items can be related by dividing the screen into smaller areas in which all associated information is contained, using lines or borders to delineate these areas. If the quantity of information is too large for any one area of associated data then one can use block scrolling which can effectivly increase the screen area without making the screen appear too cluttered.

Graphic representation of data should always be considered since it allows a greater density of data on the screen and is far easier to assimulate, the computer display is no different to other printed matter, a picture is still worth a thousand words. Having decided what is to be displayed on the screen at a particular program stage and how the information is to be presented, the screen format can be laid out on squared paper. This should be done clearly and accuratly since the layout will be used when the program is being written to determine where and when data is to be displayed. Mistakes are best corrected at this stage, they may be difficult to correct once the program is written.

A screen format consists of two parts, the first is fixed and common to all displays using the same format, and made up from borders, lines, prompts, and descriptive text. The second part consists of variable data placed on the screen by the program, this will vary each time the same format is used. When the format is being drawn out on squared paper each character space occupied by fixed display is filled in with the character which will be displayed on the screen. The character spaces occupied by variable data will simply be filled in with a crosshatch (a sample screen format layout is shown in Fig 8). Care should be taken to ensure that adiquate space is allocated to variable data otherwise the variable data may occasionaly overwrite some of the fixed data display. If variable data is occasionaly too long for the space allocated then the program must contain code to truncate the data to the maximum length displayable on the screen. All the text and graphics used in the fixed format part of the display will be embodied within the program either as direct print statements or more often as an array. Each string to be printed as part of the fixed format display will be accompanied by variables indicating the row and column number at which it is to be printed on the screen. Variables used for storing the variable section of the screen format are also accompanied by line and column values

indicating where the variable should be located on the screen. Lines, borders and other graphics will probably be generated using machine code routines, these will be called by commands written into the main program. Some of the techniques used to display a screen format will be examined in greater detail later.

THE PRINT COMMAND

The simplest way of displaying data is to use the PRINT command, this displays any specified string or numeric variable on the screen. The specified variable is placed on the screen to the right of and including the current cursor position. It is important to remember when using the PRINT command that the cursor must be in the correct position. There are two ways in which the cursor can be moved to the correct position on the screen. Firstly as the result of a previous PRINT statement, this may or may not include the use of cursor control characters. Secondly a variation of the PRINT command can be used, the TAB command, this can control the position of a string or numeric variable within a single screen line. The TAB command is simple to use providing the cursor is already positioned on the correct line, thus to print a string starting at column 20 on the current line one could use the following line of code:

```
10 PRINT TAB (20);A$
```

Further examples of the use of the TAB print command modifier are shown in the sample games program in Listing 6.

Any point on the PET screen can be discribed and located by its coordinates, these coordinates are refered to as columns in the horizontal axis and rows in the vertical axis. Thus the coordinates - column 10 row 6 - refer to a point on the screen 6 lines down from the top and 10 characters accross from the left margin (see Fig 9). The simplest method of displaying a character string at a specified location on the screen is to use the cursor control characters as part of the string. The procedure used is to have the first character of the string a cursor home, this will place the cursor at the top right corner of the screen. This is followed by cursor down characters to give the vertical coordinate of the start of the string. The number of cursor down characters used being equal to the line number on which the string is to be printed minus one, if the string to be printed is on the top line then no cursor down characters will be needed. The last block of cursor control characters are a number of

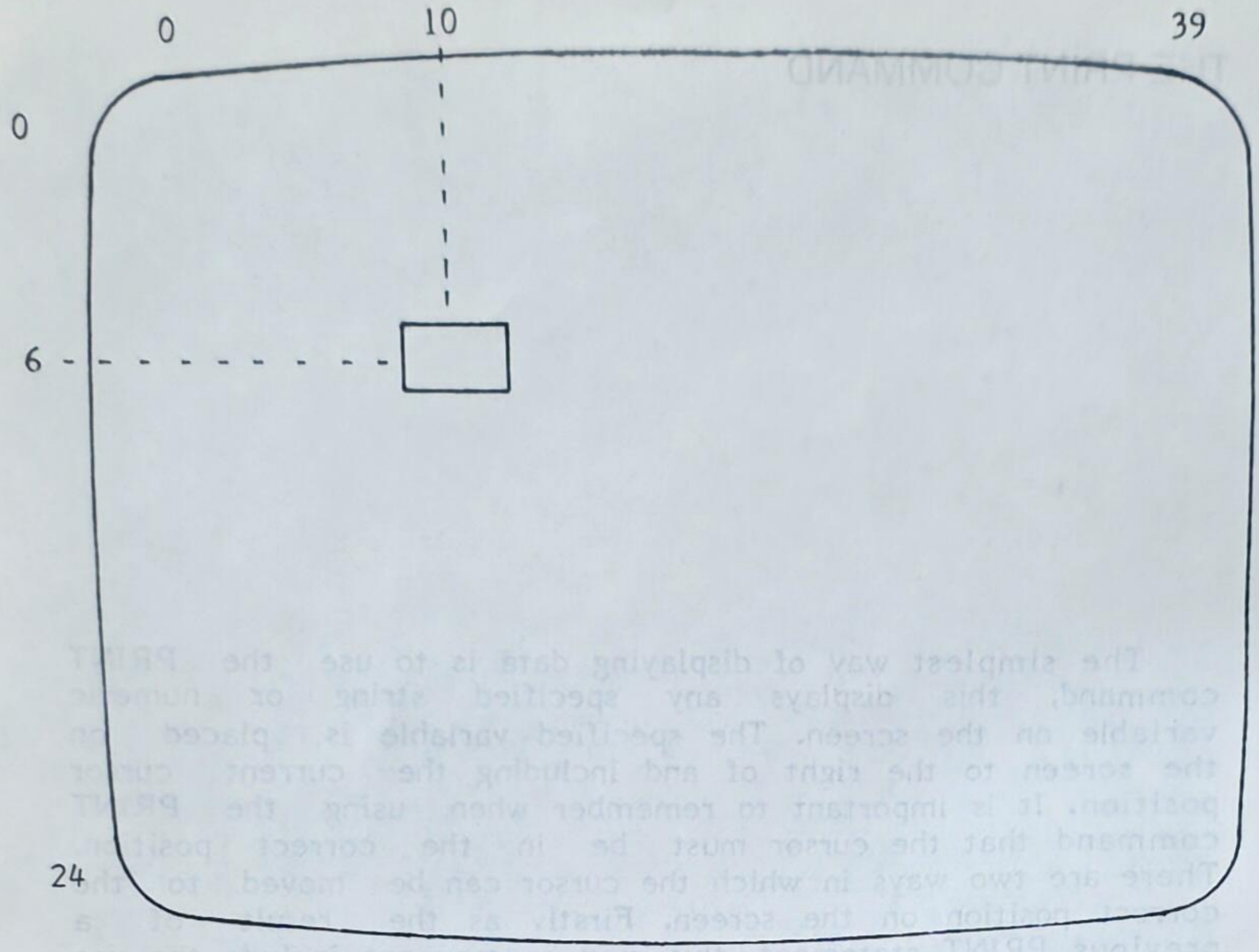


FIG 9. USE OF LINE AND COLUMN COPRDINATES

cursor right characters to give the horizontal coordinate, the number is equal to the column number minus one, no characters will be needed if the string starts at the left hand margin of the screen. An example of the use of control characters within a string is the following line of Basic:

```
30 PRINT"■■■■■■■■■■■■■■■■■■■■TEST"
```

Running this line the PET will print the string "TEST" starting at column 10 on row 6 of the screen. Though simple and very useful this method has major drawbacks, it uses a lot of memory space and lacks programming flexbility. The above example uses 17 characters (and therefor bytes of memory) to define the position at which the string is to be printed. If a lot of text is to be displayed using this method the amount of extra storage required, especialy on a small machine, could be excessive. The other disadvantage is lack of flexability, it is not possible to vary the position of a particular string once the control characters have been incorporated within the string, making it impossible to calculate the coordinates of a string on the screen.

Problems associated with using control characters incorporated within a string can be overcome by using a special subroutine to place the cursor at the desired coordinates prior to printing the string. The only

information required by the subroutine are values for the two coordinate variables, column and line. Cutting down on the amount of memory space used and giving the programmer the flexability not available when control characters are incorporated into a string. Using this method the string shown in the previous example can be printed starting on line 6 column 10 with the following line of Basic:

```
50 COL=10:LINE=6:GOSUB29800:PRINTA$
```

The subroutine called by this line is very simple, based on manipulation of two strings, one of 80 cursor right haracters and the other of 25 cursor down characters. By using the command LEFT$ a variable for the column and the string of cursor right characters a new string of cursor right characters can be created with the right number of characters to place the cursor in the correct column. Similarly using the the string of cursor down characters and the row variable a new string can be created to move the cursor down to the correct row. A program to do this is contained in Listing 1. A similar routine to the general purpose cursor control of Listing 1 is contained in the sample games program in Listing 5, this routine is modified to draw lines on the screen at specified locations.

An advantage in using the PRINT command is the ability to display strings or numeric variables of more than one character, and in the case of strings to manipulate the output using e or more of the string functions. The string function LEFT$ is used in the cursor control program to produce a string containing the correct number of cursor control characters. Similarly the string functions can be used to control the length and contents of a string prior to its being displayed. The following simple five line program gives an amusing demonstration of string manipulation using the string function MID$. Try running the program, where the input is requested type in four or five different graphics characters, the program will give a pleasing kinetic art display.

By converting numeric variables into string variables operations such as justification and rounding can be made easier, an example of this is shown in Listing 2. This program will round a number to two decimal places and then right justify it in string format by padding it with spaces.

```
10 INPUTA$
15 C$="
20 PRINT"
30 PRINTMID$(C$,INT(4*RND(1)+1),1);MID$(A$,INT(LEN(A$)*RND(1)+1),1);"
40 GOTO30
```

```
1000 REM ********************************
1001 REM *EXAMPLE OF SUBROUTINE CALL TO
1002 REM *CONTROL CURSOR POSITION PRIOR
1003 REM *TO PRINTING ON SCREEN.
1009 REM ********************************
1010 REM  **CLEAR SCREEN**
1020 PRINT"■";
1030 REM  **START PRINTING FIRST LINE AT**
1040 REM  **LINE 5 COLUMN 10**
1050 COL=10:LNE=5
1060 GOSUB 29800
1070 PRINT"*FIRST LINE COL=10,LINE=5"
1080 REM  **SECOND LINE COL=1,LINE=15**
1090 COL=1:LNE=15
1100 GOSUB29800
1110 PRINT"*SECOND LINE COL=1,LINE=15"
1500 END
2000 REM
3000 REM
4000 REM
5000 REM
6000 REM
29800 REM ********************************
29802 REM *CURSOR CONTROL SUBROUTINE
29804 REM ********************************
29806 COL$="■■■
```

```
100 SP$="                                "
1000 REM ************************************
1001 REM *EXAMPLE OF SUBROUTINE CALL
1002 REM *TO DISPLAY FORMATTED NUMBERS
1009 REM ************************************
1010 INPUT"INPUT 5 NUMBERS - ";A$(1),A$(2),A$(3),A$(4),A$(5)
1020 PRINT:PRINT:PRINT
1030 P=12
1040 FORQ=1TO5
1050 P$=A$(Q)
1060 GOSUB27000
1070 PRINTP$
1080 NEXTQ
1090 END
2000 REM
3000 REM
4000 REM
5000 REM
6000 REM
27000 REM************************************
27002 REM*SUBROUTINE TO ROUND NUMBERS
27004 REM*TO TWO DECIMAL PLACES AND
27006 REM*PAD WITH SPACES TO RIGHT
27008 REM*JUSTIFY COLUMNS. NUMBER INPUT
27010 REM*AND OUTPUT AS P$, OUTPUT
27012 REM*LENGTH P CHARACTERS.
27014 REM************************************
27016 A$="":A1$="":A2$=""
27018 AN=VAL(P$)
27020 REM
27022 REM   ***ROUND TO 2 PLACES***
27024 REM
27026 A1=-(INT(AN)-AN)
27028 A1=SGN(A1)*INT(ABS(A1*100)+.5)/100
27030 REM
27032 REM   ***FORMAT NUMBER***
27034 REM
27036 A$=STR$(A1)
27038 IFA1<.01THENA$="0.00"
27040 IFLEN(A$)=4THEN27052
27042 IFLEN(A$)>4THENA$=LEFT$(A$,4)
27044 IFLEN(A$)<4THENA$=A$+"0"
27046 REM
27048 REM   ***JUSTIFY***
27050 REM
27052 A$=RIGHT$(A$,3)
27054 A$=STR$(INT(AN))+A$
27056 P$=RIGHT$(SP$+A$,P)
27058 RETURN
READY.
```

LISTING 2.

The machine code program called in the above line is contained in Listing 3, designed as part of an integrated graphics control package it can easily be relocated anywhere in memory (provided it is protected from being overwritten by Basic). The subroutine simply uses two loops to print cursor control characters to the screen, a system subroutine call is used to to print each character. The number of times the program executes each loop depends on the values of the two coordinate variables poked into locations 84 and 85 prior to calling the subroutine. To make the program easier to use Listing 4 is a Basic loader program for the machine code cursor control routine, this can be incorporated into the begining of a program, it will load the routine into the correct part of memory and protect it. The cursor control routine can be called as in the above sample line using a SYS(31243), having previously used POKE commands to put the line number in location 84 and the column number in location 85.

LISTING 3.

```
1251  7A0B                    ;**************************************
1252  7A0B                    ;*ROUTINE TO PLACE THE CURSOR AT A
1253  7A0B                    ;*LOCATION ON THE SCREEN WHOSE COORDINATES
1254  7A0B                    ;*ARE STORED IN - $54 = LINE AND
1255  7A0B                    ;*$55 = COLUMN. CALL WITH SYS(31243)
1256  7A0B                    ;**************************************
1257  7A0B                    ;
1258  7A0B                    ;
1259  7A0B  48                CURSOR PHA
1260  7A0C  98                       TYA
1261  7A0D  48                       PHA
1262  7A0E  8A                       TXA
1263  7A0F  48                       PHA
1264  7A10  A9 13                    LDA #$13          ;ASCII HOME CHAR
1265  7A12  20 D8 E3                 JSR CDISP         ;HOME CURSOR
1266  7A15  A5 54                    LDA T1
1267  7A17  F0 09                    BEQ CUR2          ;LINE # = 0 ?
1268  7A19  A9 11             CUR1   LDA #$11           ;ASCII CURSOR DOWN
1269  7A1B  20 D8 E3                 JSR CDISP         ;DOWN CURSOR
1270  7A1E  C6 54                    DEC T1
1271  7A20  D0 F7                    BNE CUR1
1272  7A22  A5 55             CUR2   LDA T2
1273  7A24  F0 09                    BEQ CUR4          ;COLUMN #=0?
1274  7A26  A9 1D             CUR3   LDA #$1D           ;ASCII CURSOR RIGHT
1275  7A28  20 D8 E3                 JSR CDISP         ;RIGHT CURSOR
1276  7A2B  C6 55                    DEC T2
1277  7A2D  D0 F7                    BNE CUR3
1278  7A2F  68                CUR4   PLA
1279  7A30  AA                       TAX
1280  7A31  68                       PLA
1281  7A32  A8                       TAY
1282  7A33  68                       PLA
1283  7A34  60                       RTS
```

LISTING 4.

```
10 REM ************************************
20 REM *BASIC LOADER FOR MACHINE CODE
30 REM *CURSOR CONTROL ROUTINE
40 REM ************************************
100 DATA 31243
110 DATA 48,98,48,8A,48
120 DATA A9,13,20,D8,E3,A5,54,F0,09
130 DATA A9,11,20,D8,E3,C6,54,D0,F7
140 DATA A5,55,F0,09,A9,1D,20,D8,E3,C6,55,D0,F7
150 DATA 68,AA,68,A8,68,60
9000 DATA*
9010 READL
9020 READA$
9030 C=LEN(A$)
9040 IFA$="*"THEN9140
9050 IFC<1ORC>2THEN9130
9060 A=ASC(A$)-48
9070 B=ASC(RIGHT$(A$,1))-48
9080 N=B+7*(B>9)-(C=2)*(16*(A+7*(A>9)))
9090 IFN<0ORN>255THEN9130
9100 POKEL,N
9110 L=L+1
9120 GOTO9020
9130 PRINT"BYTE"L"=["A$"] ???"
9140 END
READY.
```

LISTING 5.

```
10 REM ************************************
20 REM *GAMES PROGRAM - SEE IF YOU CAN
30 REM *SUCCESSFULLY CROSS THE MINEFIELD
40 REM ************************************
50 REM SET UP MINEFIELD IN ARRAYS
60 NM=10:REM NO MINES = 10 INITIALLY
70 DIM A$(15,16)
80 GOSUB 1310
90 FOR I=1 TO 15
100 FOR J=1 TO 15
110 A$(I,J)="."
120 NEXT J
130 NEXT I
140 REM SET UP MINES AT RANDOM
150 FOR I=1 TO NM
160 X%=RND(1)*15+1
170 Y%=RND(1)*15+1
180 A$(X%,Y%)="M"
190 NEXT I
200 A$(1,1)="."
210 A$(15,15)="."
220 PRINT"[■■■■■■■■■■■■■L A N D M I N E]"
230 PRINT"[■■■■■■■■■■■■■
```

```
260 CL=15:NO=15:Z$=".":T$="H"
270 FOR RW=5 TO 19
280 GOSUB 960
290 NEXT RW
300 CL=14:NO=17:RW=4:Z$="▒":T$="H"
310 GOSUB 960
320 RW=20
330 GOSUB 960
340 CL=14:NO=17:RW=4:Z$="▒":T$="V"
350 GOSUB 960
360 CL=30
370 GOSUB 960
380 X%=1:Y%=1
390 GOTO 660
400 GOSUB 1240
410 PRINT"■";
420 GET GO$:IF GO$="" THEN 400
430 GO=VAL(GO$)
440 IF GO<1 OR GO>9 THEN 400
450 ON GO GOTO 460,480,500,520,540,550,570,590,610
460 XX%=X%-1:YY%=Y%+1:CR$="▌▒"
470 GOTO 620
480 XX%=X%:YY%=Y%+1:CR$="▒"
490 GOTO 620
500 XX%=X%+1:YY%=Y%+1:CR$="▐▒"
510 GOTO 620
520 XX%=X%-1:YY%=Y%:CR$="▌"
530 GOTO 620
540 GOTO 400
550 XX%=X%+1:YY%=Y%:CR$="▐"
560 GOTO 620
570 XX%=X%-1:YY%=Y%-1:CR$="▗▌"
580 GOTO 620
590 XX%=X%:YY%=Y%-1:CR$="▗"
600 GOTO 620
610 XX%=X%+1:YY%=Y%-1:CR$="▗▐"
620 IFXX%<1 OR XX%>15 THEN 400
630 IFYY%<1 OR YY%>15 THEN 400
640 X%=XX%:Y%=YY%
650 REM NOW CHECK FOR MINES IN VICINITY
660 IF A$(X%,Y%)="M" THEN 820
670 A$(X%,Y%)="■"
680 IF X%=15 AND Y%=15 THEN 780
690 REM CHECK FOR 1 MOVE AWAY
700 MS%=1:GOSUB 1090
710 IF MN<>0 THEN 740
720 REM CHECK FOR 2 MOVES AWAY
730 MS%=2:GOSUB 1090
740 REM RESET CURSOR WITHIN MINEFIELD
750 GOSUB 1240
760 GOTO 400
770 REM MINEFIELD CROSSED SUCCESSFULLY
780 CL=1:NO=1:RW=23:T$="H"
790 Z$="MINEFIELD SUCCESSFULLY CROSSED      "
800 GOSUB 960
810 GOTO 860
820 REM LANDMINE EXPLODES
830 CL=1:NO=1:RW=23:T$="H"
840 Z$="B A N G !!!!!!!                    "
850 GOSUB 960
```

```
860 A$(X%,Y%)="x"
870 FOR Y=1 TO 15
880 X%=1:Y%=Y:GOSUB 1240
890 FOR X=1 TO 15
900 PRINT A$(X,Y);
910 NEXT X
920 NEXT Y
930 GET GO$:IF GO$="" THEN 930
940 NM=NM+2
950 GOTO 90
960 REM SUBROUTINE TO DRAW LINE ON SCREEN
970 REM CL=START COLUMN,RW=START ROW
980 REM NO=NO OF CHARS,T$= V(VERT),H(HORIZ)
990 REM Z$=TEXT TO PRINT
1000 PRINT"■"
1010 FOR I=1 TO CL-1:PRINT"■";:NEXT I
1020 FOR I=1 TO RW-1:PRINT"■";:NEXT I
1030 IF T$="H" THEN I$=""
1040 IF T$="V" THEN I$="■■"
1050 FOR I=1 TO NO
1060 PRINTZ$;I$;
1070 NEXT I
1080 RETURN
1090 REM CHECK FOR MINES IN VICINITY
1100 REM MS%=1 CLOSE SEARCH,MS%=2 WIDE SEARCH
1110 MN=0
1120 FOR I=X%-MS% TO X%+MS%
1130 FOR J=Y%-MS% TO Y%+MS%
1140 IF I<1 OR J<1 THEN 1170
1150 IF I>15 OR J>15 THEN 1170
1160 IF A$(I,J)="M" THEN MN=MN+1
1170 NEXT J
1180 NEXT I
1190 Z$="                                                  "
1200 IF MN>0 THEN Z$=STR$(MN)+" MINES(S) "+STR$(MS%)+" MOVES AWAY"
1210 CL=1:NO=1:RW=23:T$="H"
1220 GOSUB 960
1230 RETURN
1240 REM CURSOR ADDRESS TO WITHIN MINEFIELD
1250 REM X%=X AXIS, Y%=Y AXIS
1260 PRINT"■";
1270 FOR I=1 TO X%+14:PRINT"■";:NEXT I
1280 FOR I=1 TO Y%+4:PRINT"■";:NEXT I
1290 PRINT"■■";
1300 RETURN
1310 REM INTRODUCTION & INSTRUCTIONS FOLLOW
1320 PRINT"■■■■■■■■■■■■■■■L A N D M I N E"
1330 PRINT"■■■■■■■■■■■■■■■"
1340 PRINT"THE OBJECT OF THIS GAME IS TO"
1350 PRINT"TRAVERSE A MINEFIELD. YOUR MINE"
1360 PRINT"DETECTOR GIVES SOME WARNING"
1370 PRINT"OF NEARBY MINES. BUT CAN'T"
1380 PRINT"GIVE THEIR EXACT LOCATION."
1390 PRINT"SO BE CAREFUL OR :-"
1400 PRINT"■   B A N G !!!   ■"
1410 PRINT"YOU START AT THE TOP LEFT CORNER"
1420 PRINT"AND FINISH (HOPEFULLY) AT THE                BOTTOM RIGHT."
1430 PRINT"USE THE NUMERIC PAD TO MOVE,"
1440 PRINT"WITH YOUR POSITION BEING RELATED "
```

```
 1450 PRINT"TO THE 5 KEY.■"
 1460 PRINT"          7 8 9
 1470 PRINT"           \|/
 1480 PRINT"          4-●-6
 1490 PRINT"           /|\
 1500 PRINT"          1 2 3
 1510 PRINT"■■PRESS ANY KEY WHEN READY.■"
 1520 GET GO$:IF GO$="" THEN 1520
 1530 RETURN
READY.
```

LISTING 6.

```
10 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■HANGMAN"
20 PRINT"■■■"
30 PRINT"■■          DO YOU NEED INSTRUCTIONS ?"
40 PRINT"    (REPLY Y FOR 'YES' OR N FOR 'NO')"
50 GET X$:IF X$=""THEN 50
60 IF X$="N" THEN 260
70 PRINT"■■ALWAYS REPLY Y FOR YES,OR N FOR NO."
80 PRINT"I WILL PRINT:■"
90 PRINT"                  UNKNOWN WORD=----"
100 PRINT"EACH - STANDS FOR A MISSING LETTER."
110 PRINT
120 PRINT"WHEN I PRINT '■PRESS ANY LETTER■',"
130 PRINT"FOLLOW THE INSTRUCTION !■"
140 PRINT"IF A LETTER IS IN THE UNKNOWN WORD"
150 PRINT"MORE THAN ONCE YOU WILL NEED TO PRESS"
160 PRINT"THE LETTER MORE THAN ONCE.■"
170 PRINT"E.G.'APPLE'HAS 2 P'S,PRESS P ONCE AND"
180 PRINT"I REPLY:    UNKNOWN WORD=-P---■"
190 PRINT"PRESS P A SECOND TIME AND I REPLY"
200 PRINT"                 UNKNOWN WORD=-PP--"
210 PRINT
220 PRINT"YOU HAVE TO COMPLETE THE WORD BEFORE"
230 PRINT"I HANG THE MAN."
240 PRINT"■PRESS ANY KEY WHEN READY■"
250 GETX$:IF X$=""THEN 250
260 PRINT"□";TAB(16);"HANGMAN"
270 PRINTTAB(16);"━━━━━━■";
280 RA=INT(RND(1)*197+1)
290 IF A$="END" THEN 580
300 FOR I=1 TO RA
310 READ A$:IF A$="END" THEN 580
320 NEXT I
330 RESTORE
340 LA=LEN(A$)
350 Z$=CHR$(18)+A$
360 D$=""
370 FOR I=1 TO LA
380 D$=D$+"-"
390 NEXT I
400 PRINT"■"
410 PRINTTAB(7);"UNKNOWN WORD=";D$;"■";
420 N=0:L=0
```

```
430 PRINT"■■■■■          ■PRESS ANY LETTER■  ■";
440 GET L$:IF L$=""THEN 440
450 IF L$<>" " THEN 500
460 PRINT"■■■■■   ' 'IS NOT A LETTER                "
470 GOSUB 1540
480 PRINT"■
                                                    ■";
490 GOTO 430
500 GOSUB 590
510 IF N<LA THEN 430
520 IF N=(LA+1) THEN T1=TI:GOTO 540
530 PRINT"■■■■■          YOU WIN!!                ":T1=TI
540 IF(TI-T1)<200 THEN 540
550 PRINT"■ DO YOU WANT ANOTHER GO?"
560 GET B$:IF B$=""THEN 560
570 IF B$="Y" THEN 260
580 PRINT"■ THANK-YOU FOR PLAYING":STOP
590 LF=0
600 FOR I=1 TO LA
610 B$=MID$(A$,I,1)
620 IF B$<>L$ THEN 730
630 PRINT"■■■■■          LETTER IN WORD    ■";
640 N=N+1
650 GOSUB 1540
660 IA=19+I
670 PRINT"■"
680 PRINT TAB(IA);L$;"■";
690 II=I
700 GOSUB 1390
710 LF=LF+1
720 RETURN
730 NEXTI
740 IF LF=0 THEN GOSUB 760
750 RETURN
760 L=L+1
770 PRINT"■■■■■          LETTER NOT IN WORD■"
780 GOSUB 1540
790 ON L GOSUB 890,930,1000,1040,1080,1120,1200,1280,1320
800 IF L<10 THEN RETURN
810 K=21
820 GOSUB 1500
830 PRINTTAB(19);"■■■■■
```

```
1030 RETURN
1040 K=11
1050 GOSUB 1500
1060 PRINTTAB(13);"
1070 RETURN
1080 K=7
1090 GOSUB 1500
1100 PRINTTAB(22);"
1110 RETURN
1120 K=8
1130 GOSUB 1500
1140 PRINTTAB(20);"
1150 PRINTTAB(19);"
1160 PRINTTAB(19);"
1170 PRINTTAB(20);"
1180 PRINTTAB(21);"
1190 RETURN
1200 K=12
1210 GOSUB 1500
1220 PRINTTAB(19);"
1230 PRINTTAB(18);"
1240 PRINTTAB(18);"
1250 PRINTTAB(18);"
1260 PRINTTAB(20);"
1270 RETURN
1280 K=16
1290 GOSUB 1500
1300 PRINTTAB(19);"
1310 RETURN
1320 K=17
1330 GOSUB 1500
1340 FOR I=1 TO 4
1350 PRINTTAB(20);"
1360 NEXT I
1370 PRINT"
1380 RETURN
1390 X=II-1:Y=LA-II
1400 IF X=0 AND Y=0 THEN 1450
1410 IF X=0 THEN A$=" "+RIGHT$(A$,Y):RETURN
1420 IF Y=0 THEN A$=LEFT$(A$,X)+" ":RETURN
1430 A$=LEFT$(A$,X)+" "+RIGHT$(A$,Y)
1440 RETURN
1450 A$=""
1460 FOR I=1 TO LA
1470 A$=A$+" "
1480 NEXTI
1490 RETURN
1500 FOR I=2 TO K
1510 PRINT"
1520 NEXT I
1530 RETURN
1540 T=TI
1550 IF(TI-T)<100 THEN 1550
1560 RETURN
1570 DATA ABSENCE,ACCEPT,ACCIDENTALLY
1580 DATA ACCOMMODATE,ACHIEVED,ACKNOWLEDGE
1590 DATA ACQUAINTED,ADDRESSES,AERIAL
1600 DATAAGGRAVATE,AGREEABLE,AMATEUR,AMONG
1610 DATA ANTARCTIC,ANXIETY,APPARENT,APPEARANCE
1620 DATAAPPROPRIATE,ARCTIC,ARGUMENT,ARRANGEMENTS
```

```
1630 DATA ASCENT,ATHLETIC,AWFUL,BACHELOR
1640 DATA BEGINNING,BELIEVED,BENEFITED,BREATHE
1650 DATA BRITAIN,BUSINESS,CAPTAIN,CEILING
1660 DATA CEMETERY,CERTAIN,CHOICE,CLOTHES
1670 DATA COLLEGE,COMING,COMMITTEE,COMPARATIVE
1680 DATA COMPETENT,COMPLETELY,CONSCIENTIOUS
1690 DATA CONSCIOUS,CONSISTENT,CONVENIENCE
1700 DATA COPIES,COURSE,COURTEOUS,COURTESY
1710 DATA CRITICISM,DECEIVE,DECISION,DEFINITE
1720 DATA DESIRABLE,DESPERATE,DISAPPEARED
1730 DATA DISAPPOINTED,DISASTROUS,DISCIPLINE
1740 DATA DISSATISFIED,EFFICIENCY,EIGHTH
1750 DATA ELIMINATED,EMBARRASSED,EMPHASIZE
1760 DATA ENTHUSIASM,EQUIPPED,ESPECIALLY
1770 DATA ESSENTIAL,EXAGGERATED,EXCELLENT
1780 DATA EXERCISE,EXHAUSTED,EXISTENCE
1790 DATA EXPENSE,EXPERIENCE,FAMILIAR
1800 DATA FEBRUARY,FINANCIAL,FOREIGN
1810 DATA FORMERLY,FORTY,FRIEND,GAUGE
1820 DATA GENIUS,GOVERNMENT,GRAMMAR
1830 DATA GRAMOPHONE,GRIEVANCE,GUARD
1840 DATA GUARDIAN,HANDKERCHIEF,HEIGHT
1850 DATA HEROES,HONORARY,HUMOROUS,HUNGRY
1860 DATA HURRIEDLY,HYPOCRISY,IMAGINATION
1870 DATA IMMEDIATELY,IMMIGRATE,INCIDENTALLY
1880 DATA INDEPENDENT,INDISPENSABLE
1890 DATA INFLUENTIAL,INTELLIGENCE
1900 DATA IRRESISTIBLE,KNOWLEDGE,LIGHTENING
1910 DATA LITERATURE,LIVELIHOOD,LOSE
1920 DATA LOSING,LYING,MAINTENANCE
1930 DATA MARRIAGE,MEANT,MEDICINE
1940 DATA MEDITERRANEAN,MINIATURE
1950 DATA MINUTES,MISCHIEVOUS,MURMUR
1960 DATA NECESSARY,NIECE,NOTICEABLE
1970 DATA OCCASIONAL,OCCURRED,OCCURRENCE
1980 DATA OMITTED,OPINION,OPPORTUNITY
1990 DATA ORIGINALLY,PARALLEL,PARLIAMENT
2000 DATA PASTIME,PERMANENT,PERMISSIBLE
2010 DATA PERSEVERANCE,PHYSICAL,PLANNING
2020 DATA PLEASANT,POSSESSES,PRECEDING
2030 DATA PREFERENCE,PREJUDICE,PRIVILEGE
2040 DATA PROCEDURE,PROCEEDS,PROFESSIONAL
2050 DATA PROFESSOR,PRONUNCIATION
2060 DATA PROPRIETARY,PSYCHOLOGY,QUIET
2070 DATA REALLY,RECEIVED,RECOGNIZED
2080 DATA RECOMMENDED,REFERRED,RELIEVED
2090 DATA REPETITION,RESTAURANT,RHYTHM
2100 DATA SCARCELY,SECRETARIES,SEIZE
2110 DATA SENTENCE,SEPARATE,SERGEANT
2120 DATA SEVERELY,SHINING,SIEGE,SIMILAR
2130 DATA SINCERELY,SPEECH,STRENGTH
2140 DATA SUCCESSFUL,SUPERSEDE
2150 DATA SUPPRESSION,SURPRISING,SYNONYM
2160 DATA TENDENCY,TRAGEDY,TRANSFERRED
2170 DATA TWELFTH,UNCONSCIOUS
2180 DATA UNNECESSARY,UNTIL,USUALLY
2190 DATA VALUABLE,VIEW,WEDNESDAY
2200 DATA WOOLLEN
2210 DATA END
2220 END
```

DISPLAYING CHARACTERS WITH POKE COMMANDS

Since the PET screen is part of memory, characters can be displayed randomly by poking the correct value for the character ASCII code into the desired memory location. The screen memory extends from decimal address 32768 to 33767 on a 2000 or 3000 series and 34767 on an 8000 series machine. The command POKE 33268,42 will print an asterisk in the centre of the screen on a 40 column machine. The value 42 is the ASCII code for an asterisk, it should be noted that the PET uses two different versions of the ASCII code, one used internaly within the machine and the other used by the screen circuitry, when using POKE commands to the screen it is the screen ASCII code which is used. The ASCII code values for all characters used by the PET can be obtained by consulting the table at the end of this book. The great virtue o using POKE commands to display characte s on the screen is that one is not confined, as in a PRINT statement, to displaying a sequential block of characters. POKE commands have one drawback, they require a memory address rather than the more convenient screen coordinates. This can be overcome by using a simple subroutine to convert coordinate values into screen memory addresses, thus:

```
10 REM LNE = LINE NUMBER, COL = COLUMN NUMBER
20 IFLNE <0 OR LNE >24 OR COL <0 OR COL >39 THEN END
30 Q=32768+COL+LNE*40
40 REM Q = ADDRESS FOR USE BY POKE COMMAND
```

Line 20 checks that the values for the column and line variables are within the prescribed limits for the machine (this is for a 40 column machine). Line 30 calculates the POKE address by multiplying the line number by 40 (80 for an 80 column machine) and adding it to the column value and the start of screen memory address - 32768.

When only only a few characters are to be displayed then individual POKE commands can be used for each character. An example would be a status line, where the top or bottom line of the screen is reserved for displaying the status of the program, the periferals used, etc. Such a status line might have character spaces reserved for indicating periferals in

use and any error condition. Since each character in the status line is in a fixed position and therefore with a fixed address individual POKE commands can be used. Such a line might look like this:

```
----------------------------------------------------------------
STATUS - P : D0 : D1 : ERRORS - T 00 : S 00 : D 0 : E 00
----------------------------------------------------------------
```

In this example whenever the printer file is opened the following command could be used (assuming status line is line 1 on the screen):

```
100 POKE 32777, PEEK (32777) OR 128
```

This line POKEs the screen location 32777 (where the letter P is on the status line) with the reverse field of the character currently on that line. Thus when the printer file is opened the status line will show the fact by displaying the letter P in reverse field. When the printer file is closed the following line of Basic will return the letter P to normal:

```
100 POKE 32777, PEEK (32777) AND 127
```

All the other information on the status line can be treated similarly with the correct values being placed in the error code fields.

In some cases the number of characters to be POKEd onto the screen is too large to economicaly use a seperate command for each character. In such cases the data is stored in an array or as data statements and entered onto the screen using a program loop. There are obviously many ways of doing this, the following example uses data statements with three variables for each character, the variables are column, line, and character. The variables for column and line could be replaced by an address, this would simplify and shorten the program as would storage of the character as an ASCII code rather than a character string. However using this example the data statements are much easier to write.

```
10 DATA H,V,A$,H,V,A$,H,V,A$,1000
20 READ H
30 READ V
40 READ A$
50 IF H=1000 THEN END
60 Q = 32768 + H + V * 40
70 POKE Q, ASC(A$)
80 GOTO20
```

A value of 255 for the column variable is used to indicate the end of the data statements and causes the program to exit from the loop.

Probably one of the commonest uses of POKE commands for displaying characters on the screen is the drawing of lines, bars and graphs. Horizontal lines are easily drawn using PRINT statements but vertical lines are best drawn using POKE commands, the following statement will draw a vertical line from top to bottom of the screen:

```
100 FOR X = 32780 TO 33580 STEP 40: POKE X , 101 : NEXT X
```

A diagonal line starting at the same origin and at an angle of 45 degrees can be drawn with the following statement:

```
100 FOR X = 32780 TO 33580 STEP 41: POKE X , 77 : NEXT X
```

A triangle can be drawn (it should be noted that the PET graphics characters limit one to angles of 45, 90 or 135 degrees, other angles can only be drawn by having steps in a line)by a combination of three such single line character POKEing loops.

```
100 FOR X = 32780 TO 33560 STEP 41: POKE X, 77: NEXT X
110 FOR X = 32779 TO 33540 STEP 40: POKE X, 89: NEXT X
120 FOR X = 33580 TO 33599 : POKE X, 69: NEXT X
```

The screen addresses used in each loop are the location of the top character in the line and the location of the bottom character in the line. The step increment in the line determines whether the line is vertical, horizontal, or an angle to the left or an angle to the right (all values displayed are for a 40 column machine, for the 80 column add 40 to all the step increments except the horizontal line increment which stays at 1):

```
STEP 40    = vertical line
STEP 1     = horizontal line
STEP 39    = angle from right to left
STEP 41    = angle from left to right
```

The ASCII value used in the POKE command depends on the application and the table of ASCII codes for the various PET graphics characters - note it is the screen ASCII code which is required not the standard internal ASCII code. Borders are commonly required in programs either simply to make the display look neat or to divide the screen into different sections. A border is easily and quickly drawn using a subroutine consisting of four line drawing loops and four individual pokes to add the corner characters for a thin line border, the characters used for the border can be varied by changing the ASCII code values. The size and position of the border can be varied by changing the addresses used in each loop. The example, program Listing 7, will draw a border around the outside edge of a 40 column screen, by reducing the second address in each of the horizontal loops by 20 and subtracting 20 from the starting and ending addresses of the

right hand vertical line loop the border can be drawn around the left hand half of the screen. When designing a screen display using borders and lines it is best to lay it out on a peice of graph paper with one square for each character this will make it much easier to determine the exact starting and ending addresses of each line. An example of a routine to draw a fairly complex shape is shown at the begining of the program in Listing 8.

LISTING 7.

```
5 REM ************************************
6 REM *DRAW VARIABLE SIZE BORDER, Z1=COLUMN START
7 REM *Z2=COLUMN END;W1=LINE START; W2=LINE END
8 REM ************************************
10 PRINT"□";
20 Z1=10:Z2=15:W1=10:W2=20
30 BT=W1*40+Z1+32727
40 BL=Z2-Z1
50 BH=W1-W2
60 BB=W2*40+Z1+32767
70 POKEBT,79:POKEBT+BL,80
80 POKEBB,76:POKEBB+BL,122
90 FORQ=BB+1TOBB+BL-1:POKEQ,100:NEXTQ
100 FORQ=BT+40TOBB-40STEP40:POKEQ,101:NEXTQ
110 FORQ=BT+40+BLTOBB+BL-40STEP40:POKEQ,103:NEXTQ
120 FORQ=BT+1TOBT+BL-1:POKEQ,99:NEXTQ
130 END
```

The POKE command can also be used for reverse fielding or giving a background character to areas of the screen, this is frequently used to highlight areas where important data is being displayed. Reverse fielding a character using POKE was looked at earlier on in the example of a status line, to reverse field an area on the screen simply requires a loop program.

```
10 A=32980:LNE=7:COL=10
20 FORX=1TOLNE
30 FORQ=1TOCOL
40 POKEA+Q,PEEK(A+Q)OR128
50 NEXTQ
60 A=A+40
70 NEXTX
READY.
```

By changing the start address or the values for the height or width the reverse field block can be moved around the screen to any desired position or changed in size and dimensions. A slight variation on this program can be used to give a

background character block, characters printed in this block either ordinary or in reverse field stand out much better than against an ordinary dark background.

```
10 A=32980:LNE=7:COL=10
20 FORX=1TOLNE
30 FORQ=1TOCOL
40 POKEA+Q,230
50 NEXTQ
60 A=A+40
70 NEXTX
READY.
```

The ability to move a character across the screen finds many applications, particularly in games programs, a ball bouncing between two bats on the screen or a photon torpedo fired from a starship. Using PEEK and POKE commands moving a single character in straight lines, either horizontal,vertical or at one of a range of different angles, across the screen is very easy. The following six line program will move a star verticaly up and down the left hand side of the screen (column 1) of a 40 column machine. By incrementing he starting address in line 10 from 32768 to 32778 and the ending address to 33857 the asterisk can be made to move verticaly in column 10 and by changing the ending address one can change the number of lines moved by the asterisk. Line 30 is simply a delay to stop the asterisk moving too fast across the screen. Other characters can be made to move across the screen by changing the ASCII code value used in line 20, line 40 replaces the character displayed by a space after a slight pause generated in line 30.

```
10 FORQ=32768 TO 33767 STEP 40
20 POKE Q, 42
30 FOR X=1 TO 10: NEXT X
40 POKEQ,32
50 NEXT Q
60 GOTO10
```

The following program is very similar except that it moves the asterisk horizontaly across line 1, incrementing the starting and ending addresses by 40 will move the line down one in a 40 column machine.

```
10 FORQ=32768 TO 32807
20 POKE Q, 42
30 FOR X=1 TO 10: NEXT X
40 POKEQ,32
50 NEXT Q
60 GOTO10
```

To move a character in a diagonal line use the same method as

used to move in a vertical line, except that the step increment in line 10 is greater or less than 40, the angle of the line can be changed by changing the increment.

```
10 FORQ=32768 TO 33500 STEP 42
20 POKE Q, 42
30 FOR X=1 TO 10: NEXT X
40 POKEQ,32
50 NEXT Q
60 GOTO10
```

In all three of the above example programs it was assumed that the character is moving across a blank screen thus any existing characters on the screen will, if they lie in the path of the moving character be replaced by a space. This in many applications is undesirable. The following modified peice of code allows a moving character to pass through an existing display leaving it unaltered.

```
10 FORQ=32768 TO 33500 STEP 42
15 A = PEEK(Q)
20 POKE Q, 42
30 FOR X=1 TO 10: NEXT X
40 POKEQ,A
50 NEXT Q
60 GOTO10
```

Characters in the previous examples all move from left to right or top to bottom, they can be made to move in the opposite direction by changing the loop used in line 10 so that the address value goes from a high value to a lower value with a negative step, thus.

```
10 FORQ=33767 TO 33000 STEP -42
15 A = PEEK(Q)
20 POKE Q, 42
30 FOR X=1 TO 10: NEXT X
40 POKEQ,A
50 NEXT Q
60 GOTO10
```

Note: all the above routines are for 40 column machines, to use on an 80 column machine add 40 to the step increment except for the horizontal which stays unchanged.

LISTING 8.

```
10 REM ************************************
20 REM *GAME TO DRIVE A CAR AROUND A RACE TRACK
30 REM *USE THE NUMBER KEYS TO CONTROL SPEED
40 REM *BRAKING AND GEAR CHANGES.
50 REM *TRY NOT TO TAKE THE BENDS TOO FAST YOU
60 REM *MAY SKID OFF THE TRACK - BEST TIME WINS.
70 REM ************************************
80 DIM A1(14),B1(14),C1(14),GS(5),BS(10),MS(14)
90 X=32859:G=1:R=0:S=0
100 PRINT"■
110 FOR Z=1TO14:READ A,B,C:A1(Z)=A:B1(Z)=B:C1(Z)=C
120 FOR D=1TO A:POKE X,42:X=X+B:NEXT D
130 X=X+C
140 NEXT Z
150 DATA 20,1,40,3,41,-40,4,1,40,2,41,-1,2,40,-1,6,-1,-40,3,-41,40
160 DATA 2,39,1,2,39,-40,4,-1,-40,5,-41,40,12,-1,-40,2,-41,2,6,1,1
170 FOR Z=1TO5:READ GS(Z):NEXT
180 DATA 2.4,1.9,1.5,1.2,1
190 FOR Z=0TO9:READ BS(Z):NEXT
200 DATA 0,6,15,20,25,30,40,59,80,98
210 FOR Z=1TO14:READ MS(Z):NEXT
220 DATA 70,70,70,85,70,70,35,40,60,60,70,80,40,200
230 D=10:GOSUB 890
240 PRINT"7 TO CHANGE UP
250 PRINT"4 TO CHANGE DOWN
260 PRINT"8 TO BRAKE
270 PRINT"9 TO ACCEL
280 PRINT"S TO START
290 GET S$:IF S$="" THEN 290
300 X=32859:Y=X:S9=TI
310 A=3:GOSUB 860
320 FOR Z=1 TO 14
330 A2=A1(Z):B2=B1(Z):C2=C1(Z):MX=MS(Z)
340 D=15:GOSUB 890
350 IF Z<14 THEN PRINT"MAX SPEED AT NEXT BEND IS ";MX
360 FOR DL=1TO A2
370 D=16:GOSUB 890
380 PRINT"GEAR=";G;"REVS=";INT(R);"MPH=";INT(S);"■■      "
390 FORZZ=1TO500:NEXT
400 POKE X,145:POKE Y,42:Y=X:X=X+B2
410 GET N
420 IF N=0 AND NN=0 THEN 500
430 IF N=0 AND NN=1 THEN NN=0:GOTO 490
440 IF N=7 THEN G=G+1:IF G>5 THEN G=4:GOTO 620
450 IF N=4 THEN G=G-1:IF G<1 THEN G=1:GOTO 620
460 IF N=8 OR N=5 OR N=2 THEN GOSUB 770
470 IF N=9 THEN GOSUB 830
480 NN=1:GOTO 410
490 GOSUB 860
500 IF S<2 THEN 590
510 IF R<2000 THEN 580
520 IF R=<12000 THEN 560
```

```
530 IF R>14000 THEN 610
540 OV=OV+1:IF OV>2 THEN 610
550 D=17:GOSUB 890:PRINT"■WARNING■ YOU HAVE OVERREVED":S=S-S/4
560 NL=1:IF S<200 THEN NL=INT((200-S)/8)
570 GOSUB 750:D=18:GOSUB 890:GOTO 630
580 D=17:GOSUB 890:PRINT"■YOU HAVE STALLED■":GOTO 600
590 D=17:GOSUB 890:PRINT"■YOU HAVE BRAKED TO A STANDSTILL■
600 B=0:G=1:S=0:A=3:GOSUB 860:GOTO560
610 D=17:GOSUB 890:PRINT"■YOU HAVE BLOWN UP":END
620 D=17:GOSUB 890:PRINT"■CHECK YOUR GEARS":GOTO 560
630 NEXT DL
640 X=X+C2
650 IF S<MX OR Z=14 THEN 690
660 IF S>(MX+MX/5) THEN D=17:GOSUB 890:PRINT"■YOU HAVE CRASHED■":END
670 D=17:GOSUB 890:PRINT"■YOU HAVE SKIDED & LOST SPEED■"
680 S=S-S/3
690 NEXT Z
700 TS=(TI-59)/60:TM=INT(TS/60)
710 TS=TS-(TM*60)
720 D=17:GOSUB 890:
730 PRINT"YOUR LAP TIME WAS";TM;"MINS";TS;"SECS
740 END
750 FOR L=0TONL:L1=SQR(NL):NEXT
760 RETURN
770 IF N=8 THEN VV=3
780 IF N=5 THEN VV=2
790 IF N=2 THEN VV=1
800 B=B+VV:A=0
810 IF B>9 THEN B=9
820 RETURN
830 A=A+1:IF A>9 THEN A=9
840 B=0
850 RETURN
860 S=S+(A*3.2*GS(G))-BS(B)-S/20
870 R=S*GS(G)*61
880 RETURN
890 PRINT"■
900 IF D>16 THEN D=D-1:ML=1
910 FOR D1=1 TO D
920 PRINT
930 NEXT
940 IF ML=0 THEN 960
950 ML=0:PRINT"                                        "
960 RETURN
970 END
READY.
```

DISPLAYS USING MACHINE CODE ROUTINES

Using machine code routines to perform specific display functions gives the programmer a far wider range of options than could be practicaly obtained with Basic. Machine code routines have considerable advantages over comperable routines written in Basic, they are very much faster, use less memory, and are more flexable. However, they are also much harder to write and understand especialy if your knowledge of machine code programming is negligable. An understanding of machine code is not required to use the routines in this book, although a slight knowledge will add interest.

All the machine code routines are designed to be integrated as a complete package which sits at the top of memory - all programs are written for a 32K 40 column machine (the package is available on disk from the publishers, this also includes versions for other CBM machines and all example programs in the book). This package should be loaded prior to loading the Basic program thus for a 3032 PET one would use the following sequence:

1) ..... Load machine code program GRAPHICS from disk and — RUN.

2) ..... Load the Basic program which will use the machine code routines. The first line of this program must perform the function of lowering the top of memory to protect the machine code programs. The following line of code will do this for all programs using routines other than the large screen routines which require data storage space in excess of that normal allocated to the package. The section on large screen displays includes details on the correct lowering of memory values for use in line one of the calling program.

```
1 POKE48,0:POKE49,112:POKE52,0:POKE53,112:CLR
```

Note: the machine code routines will stay in the machine until it is switched off and will be untouched by the NEW command or the loading of further Basic programs.

The package could of course be relocated and placed on a

ROM, this would be more convenient for the user but would not offer the flexability of a softloaded program, which being located in RAM is easily user modifiable.

The procedure for using a machine code subroutine within a program is similar to that used when calling a subroutine written in Basic. Firstly parameters required by the subroutine must be passed from the main Basic program, by storing the parameter values in specific memory locations using POKE commands. Secondly the subroutine is called using a SYS command with the appropriate starting address of the routine. Once the routine has been executed control will return to the Basic program which will continue with the next line of Basic code. Any parameters returned by the subroutine can be accessed from specific memory locations using PEEK commands. Consider the following example, we wish to draw a horizontal bar of 15 reverse field space characters starting on line 10 and column 12 (the horizontal line being drawn from left to right). The following Basic code would be used to pass these parameters for use by the subroutine (note the commands have been put on seperate lines for clarity in practice they would be on the same line and seperated by colons):

```
100 POKE 0,160      :place ASCII code for RVS space in location 0
110 POKE 86,10      :starting column number placed in location 86
120 POKE 87,10      :starting line number placed in location 87
130 POKE 88,15      :line length (number of characters) in location 88
```

Having passed the required parameters (these could of course be variables rather than fixed values) and placed them in the correct memory locations used by that particular subroutine the routine may be called:

```
140 SYS(30029)      :jump to horizontal bar drawing subroutine
```

If variables are used for the column and line then we can check that they are within the limits imposed by screen size - 40 columns and 25 lines maximum on a 40 column machine. This is done by looking at a parameter passed from the machine code routine and stored in location 1. If the value in location 1 is 0 then those values are within bounds and therefore OK, if the value is 1,2,3 or 4 then they are outside the limits and have defaulted to the maximum permissable values. Each of the 30 diff ent subroutines within the package have their own specific parameters and parameter locations together with a unique starting address, these are explained in full in the text and examples accompanying each subroutine.

General features of routines within the package.

The machine code functions in the package utilise the fact that the PET has a memory mapped video display. All the routines are thus based to a greater or lesser degree on the movement of data within a block of memory - the screen and temporary screen buffer memory. Since these routines have to accept data from a Basic program in a form easily interpreted by a user, the majority, use a common subroutine to convert a value for screen line, and column number, into the screen memory address. This routine is shown in Listing 1, it is not intended to be called from a SYS command and any attempt to do so will probably cause the machine to crash. The subroutine converts the column number in location 86 and line number in location 87 into the two byte address of the screen memory location lying at these coordinates. The routine also checks that the values given are within the limits for the machine, attempts to use values of less than zero, line numbers greater than 25, or column numbers greater than 40, will set an error flag. The error flag is in location 1 and normaly set to 0, if an error condition exists one of four error codes is used:

1 .... column number less than 0
2 .... column number greater than 40
3 .... line number less than 0
4 .... line number greater than 40

These error codes are common to all routines in the package which utilise Listing 1 to convert column and line number to screen memory address. Since this routine is used by many of the subroutines in the machine code graphics package, the locations for storage of the line and column parameters are identical in these routines. The majority of subroutines in the package share memory locations for storage of parameters, all such locations are assigned names, these are shown in Table 1. The use of subroutines called by more than one function routine within the package has been deliberatly kept to a minimum to allow the independent loading of each routine. In some situations the programmer may only wish to use one or two functions in the package and be unwilling to lose the memory space required by the whole package. In these situations the individual routines plus ADR1 (if used) can be loaded and relocated to the top of memory, this is made much easier by keeping the number of subroutines used by the package to a minimum.

The routines in the package have been written in a way which does not give optimal memory usage but which does give reasonable clarity, so that readers with little previous experience of machine code will be able to follow the logic flow easily. To facilitate this each function routine has been written as a stand alone program with at most just a single call to an external subroutine, subroutine calls within each function routine have also been kept to a

minimum. The machine code package has been written using the Commodore 6502 Assembler and the listings in this book are all in standard assembler format, the following is a brief description, full information will be found in the assembler manual. An assembler listing is divided into nine vertical columns thus:

```
1257   7A12   20 D8 E3                JSR CDISP       ;HOME CURSOR
1258   7A15   A5 54                   LDA T1
1259   7A17   F0 09                   BEQ CUR2        ;LINE # = 0 ?
1260   7A19   A9 11          CUR1     LDA #$11         ;ASCII CURSOR DOWN
```

When interpreting a listing each of these nine columns has a particular meaning these can be sumarised as follows:
1 .... this is the line of the assembled listing and simply refers to the order in which the routines were written, when interpreting a particular peice of code it can be ignored.
2 .... this column is important since it contains the memory address in hexadecimal of the first byte of the instruction code contained on that same line. Each instruction can occupy between one and three bytes of memory.
3 .... this column contains the first byte of the instruction code on that line, the byte is given in hexadecimal notation, this byte is located at the address in column 2 (column 3 contains the object code instruction).
4 .... the second byte of the instruction code (columns 4 and 5 are the object code address of the instruction code).
5 .... the third byte of the instruction code.
6 .... this column contains the jump address label which is used by the assembler to identify the line to which control jumps as the result of a conditional or non conditional jump command else where in the program, it is analagous to the line number used in the Basic commands GOTO or GOSUB.
7 .... is the mnemonic for the opcode instruction (its hexadecimal value is contained in column 3) thus LDA is a shorthand description for the instruction - LoaD Accumulator with contents of the operand location in column 8. These are the standartd mnemonics for all the instructions in the 6502 instruction set and are recognised by the assembler during assembly when they are assigned the numerical value required by the processor.
8 .... the operand required by the opcode instruction is put in this column, the operand can be either a value or a label, the label referencing either a memory location where data is stored or a jump address. All values contained in the operand column are stored in hexadecimal notation. All operand labels which referance locations where data is stored are given in Table 2 together with the associated address where that data is stored. Operand labels referencing jump addresses are shown together with the address of the jump location in Table 1.
9 .... this is the comments column and is used to give a written descrip on of the function of each section of code, it's function is the same as the REM statement in Basic. By providing plenty of comments the user can determine the exact

function of a section of code without having to decipher the actual code.

TABLE 1.

```
0011  0000
0012  0000                ;
0013  0000                ;VARIABLES
0014  0000                ;
0015  0000                T8=$00
0016  0000                T9=$01
0017  0000                T1=$54
0018  0000                T2=$55
0019  0000                T3=$56
0020  0000                T4=$57
0021  0000                T5=$58
0022  0000                T6=$59
0023  0000                T7=$5A
0024  0000                MCOL=$033A
0025  0000                MLINE=$033B
0026  0000                TOTCOL=$033C
0027  0000                LSTART=$033D
0028  0000                CCOL=$033E
0029  0000                CLINE=$033F
0030  0000                BINOFF=$0340
0031  0000                XD=$0341
0032  0000                YD=$0342
0033  0000                A0=$0343
0034  0000                A1=$3044
0035  0000                YE=$0345
0036  0000                XE=$0346
0037  0000                D1=$0347
0038  0000                S0=$0348
0039  0000                S1=$0349
0040  0000                LG=$034A
0041  0000                SH=$034B
0042  0000                TT=$034C
0043  0000                TS=$034D
0044  0000                UD=$034E
0045  0000                CT=$034F
0046  0000                XCOORD=$0350
0047  0000                YCOORD=$0351
0048  0000                AORD=$0352
0049  0000                X1=$0353
0050  0000                Y1=$0354
0051  0000                X2=$0355
0052  0000                Y2=$0356
0053  0000                ARD=$0357
0054  0000                ERROR=$0358
0055  0000                ERRORL=$0359
0056  0000                HB=$0360
0057  0000                WB=$0361
0058  0000                ;
0059  0000                ;
0060  0000                CDISP=$E3D8
0061  0000                ;
0062  0000                ;START ADDRESS OF PACKAGE
0063  0000                ;
0064  7400                      *=$7400
                          ;
```

LISTING 1.

```
LINE# LOC    CODE          LINE

0066  7400                ;**********************************
0067  7400                ;* ROUTINE TO CALCULATE SCREEN ADDRESS
0068  7400                ;* FROM VALUES FOR COLUMN AND LINE
0069  7400                ;* COLUMN IN $56, LINE IN $57
0070  7400                ;* ADDRESS RETURNED IN $54 AND $55.
0071  7400                ;* THIS ROUTINE IS ONLY CALLED BY OTHER
0072  7400                ;* ROUTINES IN THE PACKAGE, DO NOT
0073  7400                ;* USE BY ITSELF IT WILL CAUSE PET
0074  7400                ;* TO CRASH SINCE REGISTERS ARE NOT
0075  7400                ;* SAVED ON STACK.
0076  7400                ;* ERROR FLAG IN LOCATION $01
0077  7400                ;**********************************
0078  7400                ;
0079  7400  A9 00    ADR1    LDA #$00
0080  7402  85 01            STA T9
0081  7404  A5 56            LDA T3
0082  7406  30 33    ADR2    BMI ERROR1        ;COLUMN < 0 ?
0083  7408  C9 28            CMP #$28
0084  740A  B0 3A            BCS ERROR2        ;COLUMN > 40 ?
0085  740C  A5 57            LDA T4
0086  740E  30 41            BMI ERROR3        ;LINE < 0 ?
0087  7410  C9 19            CMP #$19
0088  7412  B0 48            BCS ERROR4        ;LINE > 25 ?
0089  7414  A9 00            LDA #$00
0090  7416  85 54            STA T1
0091  7418  85 55            STA T2
0092  741A  A5 57            LDA T4
0093  741C  F0 0F            BEQ COLUMN        ;LINE = 0
0094  741E  AA                TAX
0095  741F  18        LINE    CLC
0096  7420  A5 54            LDA T1
0097  7422  69 28            ADC #$28          ;ADD 40 TO ADDRESS
0098  7424  85 54            STA T1
0099  7426  90 02            BCC LINE2
0100  7428  E6 55            INC T2
0101  742A  CA        LINE2   DEX
0102  742B  D0 F2            BNE LINE          ;ANOTHER LINE ? DO AGAIN
0103  742D  18        COLUMN  CLC
0104  742E  A5 54            LDA T1
0105  7430  65 56            ADC T3            ;ADD COLUMN # TO ADDRESS
0106  7432  85 54            STA T1
0107  7434  A5 55            LDA T2
0108  7436  69 80            ADC #$80          ;ADD SCREEN STARTING ADD
0109  7438  85 55            STA T2
0110  743A  60                RTS
0111  743B                ;
0112  743B                ;
0113  743B                ;
0114  743B  A9 01    ERROR1  LDA #$01           ;SET ERROR FLAG
0115  743D  85 01            STA T9            ;STORE IN 01
0116  743F  A9 00            LDA #$00
0117  7441  85 56            STA T3            ;SET COLUMN TO 0
0118  7443  4C 06 74          JMP ADR2          ;CALCULATE ADDRESS
0119  7446  A9 02    ERROR2  LDA #$02
0120  7448  85 01            STA T9
```

```
LINE# LOC    CODE                LINE

0121   744A   A9 27                    LDA #$27        ;SET COLUMN TO 40
0122   744C   85 56                    STA T3
0123   744E   4C 06 74                 JMP ADR2
0124   7451   A9 03          ERROR3    LDA #$03
0125   7453   85 01                    STA T9
0126   7455   A9 00                    LDA #$00        ;SET LINE TO 0
0127   7457   85 57                    STA T4
0128   7459   4C 06 74                 JMP ADR2
0129   745C   A9 04          ERROR4    LDA #$04
0130   745E   85 01                    STA T9
0131   7460   A9 18                    LDA #$18        ;SET LINE TO 25
0132   7462   85 57                    STA T4
0133   7464   4C 06 74                 JMP ADR2
0134   7467                ;
0135   7467                ;
```

Bytes = 102.

TABLE 2.

SYMBOL TABLE

SYMBOL	VALUE						
A0	0343	A1	3044	ADDPT	7B17	ADR1	7400
ADR2	7406	AORD	0352	ARD	0357	BCHAR	7A35
BCHAR1	7A49	BCHAR2	7A4C	BCHAR3	7A58	BCHAR4	7A5E
BCHAR5	7A61	BCHAR6	7A64	BCHAR7	7A77	BCHAR8	7A82
BINOFF	0340	BLOCK	7706	BLOCK1	7710	BLOCK2	7713
BLOCK3	7725	BORD	758A	BORD1	75DD	CCOL	033E
CDISP	E3D8	CLINE	033F	COLUMN	742D	CT	034F
CUR1	7A19	CUR2	7A22	CUR3	7A26	CUR4	7A2F
CURSOR	7A0B	D1	0347	DOPNT	7C55	DPAG1	79E5
DPAGE	79D7	DSCRL1	77B0	DSCRL2	77BB	DSCRL3	77CD
DSCRL4	77D0	DSCRL5	77E5	DSCRL6	77F0	DSCRL7	77F3
DSCRL8	77F6	DSCROL	77A5	ENDEXP	7AF9	ENDP	7B34
ER	7CA3	ERASPT	7B1F	ERROR	0358	ERROR1	743B
ERROR2	7446	ERROR3	7451	ERROR4	745C	ERRORL	0359
EXP	7AEF	FOUND	7B12	HB	0360	HBAR	754D
HBAR2	7557	HBORD	75FA	HBORD1	75FC	HOR2	7471
HOR3	7486	HOR4	7489	HOR5	7495	HORIZ	7467
INVT	772E	INVT1	7738	INVT2	773B	INVT3	774F
LADR1	788C	LADR2	7890	LADR3	78AD	LADR4	78B9
LADR5	78BC	LCLR	7928	LCLR1	792D	LCLR2	7936
LCLR3	793A	LDISP	794D	LDISP0	795C	LDISP1	7969
LDISP2	796B	LDISP3	797E	LDISP4	7989	LER1	78CB
LER2	78D9	LER3	78E8	LER4	78F6	LG	034A
LHOR2	7629	LHOR3	763E	LHOR4	7641	LHOR5	764D
LHORIZ	761F	LINE	741F	LINE2	742A	LINIT	7905
LPLOT	7B4A	LRHOR1	7692	LRHOR2	769C	LRHOR3	76B1
LRHOR4	76B4	LRHOR5	76C0	LSAVE	7992	LSAVE0	79A1
LSAVE1	79AE	LSAVE2	79B0	LSAVE3	79C3	LSAVE4	79CE
LSCRL1	785E	LSCRL2	7860	LSCRL3	7878	LSCRL4	7883
LSCROL	7847	LSTART	033D	LTVER1	76D3	LTVER2	76F0
LTVER3	76FC	LTVERT	76C9	LVER1	7660	LVER2	767D
LVER3	7689	LVERT	7656	MCOL	033A	MLINE	033B
MOREC	7B00	NEWC	7B2C	PPOINT	7A8B	RHOR1	74DA
RHOR2	74E4	RHOR3	74F9	RHOR4	74FC	RHOR5	7508
RSCRL1	781A	RSCRL2	781D	RSCRL3	7833	RSCRL4	783E
RSCROL	7803	RTN	7CA1	S0	0348	S1	0349
S2030	7B54	S2100	7B84	S2110	7B8E	S2130	7B98
S2140	7BA0	S2150	7BAF	S2170A	7BAA	S2170B	7BBC
S2170C	7BC1	S2180	7BCE	S2190	7BD0	S2200	7BE6
S2210	7BF0	S2220	7BF3	S2230	7C09	S2250	7C13
S2255	7C3F	S2255A	7C48	S2255B	7C50	S2270	7C33
S2280	7C5B	S2290	7C60	S2310	7C7B	S2320	7C7E
S2360	7C99	S2370	7C9C	S2380	7C9E	SH	034B
SORIG	7ABA	SPAG1	79FF	SPAGE	79F1	T1	0054
T2	0055	T3	0056	T4	0057	T5	0058
T6	0059	T7	005A	T8	0000	T9	0001
TABLE	7B3A	TESTX2	7B60	TESTY1	7B6C	TESTY2	7B78
TOTCOL	033C	TS	034D	TT	034C	TVER1	751B
TVER2	7538	TVER3	7544	TVERT	7511	UD	034E
USCRL1	776F	USCRL2	7772	USCRL3	7787	USCRL4	7792
USCRL5	7795	USCRL6	7798	USCROL	7758	VBAR	7564
VBAR2	7570	VBORD	7604	VBORD1	760A	VER1	74A8
VER2	74C5	VER3	74D1	VERT	749E	WB	0361
X1	0353	X2	0355	XCOORD	0350	XD	0341
XE	0346	XOK	7AB2	Y1	0354	Y2	0356
YCOORD	0351	YD	0342	YE	0345	YOK	7AA8

2 SIMPLE GRAPHICS DISPLAY ROUTINES

Section 2

Simple machine code graphics display routines - 45
Draw horizontal bar of specified character - 46
Draw vertical bar of specified character - 48
Fill block of screen with specified character - 50
Reverse field block of screen - 52
Draw border around specified screen area - 54
Basic program showing application of above routines - 57

```
DESCRIPTION          | QTY | COST | TOTAL
                     |     |      |
                     |     |      |
                     |     |      |
```

SIMPLE GRAPHICS DISPLAY ROUTINES 2

The five routines in this section are all very simple and provide an introduction to the basic methods employed in the more sophisticated routines comprising the rest of the package. All rely on the initial calculation of the screen memory address of the first character in the line or block to be displayed, this is usualy the top left corner character of that line or block. The address is calculated from the line and column numbers which designate the coordinates of the character using the routine ADR1 starting at location hex $7400. All the display functions are performed by the calculation of screen memory addresses using the starting address as a reference point. The 1000 bytes of screen memory in a 40 column machine can be subdivided into 25 smaller blocks (lines) each of 40 locations. Drawing a line from top to bottom of the screen in column 5 is simply a matter of filling every 5th location in each of the 25 sub blocks with the desired ASCII code value. Drawing a horizontal line accross the screen simply means filling all the locations in one line block of memory with the specified character. The starting position within a sub block is determined by the starting screen address and the length of the line is determined by a variable transfered from the Basic program. Filling a block of the screen with a specified character simply means repeatedly drawing horizontal lines until the block is filled. To reverse field a line or block of the screen the character in each screen memory location has a logical Exclusive OR operation performed upon it.

Horizontal Bar.

The routine HBAR will draw a horizontal bar on the screen with a single character space resolution. The position of the left hand end of the bar is determined by two variables - column number and line number. The length of the bar can be any integer value with a minimum of 1 and a maximum of 255 character spaces, in practice the maximum line length will be equal to the screen width. A variable allows the character used to draw the line to be defined in the Basic calling program. The variable locations required by the routine are as follows:

86 .... column number of horizontal line start (left end).
87 .... screen line number of horizontal line start.
88 .... length of horizontal line.
00 .... ASCII code value of characters used in horizontal line.

The routine is called from a Basic program with the command:

SYS(30029)

Some example applications for this function:
Underlining important data.
Demarcation between text or data areas.
To erase lines by using the ASCII space code in 00.
In drawing graphic diagrams.

The following is an example of a Basic program using this routine:

```
10 POKE 86,5  : REM START AT COLUMN # 5
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,10 : REM LINE LENGTH 10 CHARACTERS
40 POKE 0,160 : REM USE RVS SPACE CHARACTERS FOR LINE
50 SYS(30029) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0309  754D                ;*************************************
0310  754D                ;* DRAW A HORIZONTAL BAR OF ANY PET
0311  754D                ;* CHARACTER. CHARACTER STORED IN $00
0312  754D                ;* COLUMN START IN $56, LINE OF START
0313  754D                ;* IN $57, LENGTH OF LINE IN $58.
0314  754D                ;* ROUTINE CALLED BY - SYS(30029)
0315  754D                ;*************************************
0316  754D                ;
0317  754D                ;
0318  754D  48           HBAR    PHA
0319  754E  98                   TYA
0320  754F  48                   PHA
0321  7550  8A                   TXA
0322  7551  48                   PHA
0323  7552  20 00 74              JSR ADR1
0324  7555  A4 58                LDY T5
0325  7557  A5 00         HBAR2   LDA T8
0326  7559  91 54                STA (T1),Y
0327  755B  88                   DEY
0328  755C  D0 F9                BNE HBAR2
0329  755E  68                   PLA
0330  755F  AA                   TAX
0331  7560  68                   PLA
0332  7561  A8                   TAY
0333  7562  68                   PLA
0334  7563  60                   RTS
0335  7564                ;
```

Bytes = 23

Vertical Bar.

The routine VBAR will draw a vertical bar on the screen with a single character space resolution. The position of the top end of the bar is determined by two variables - column number and line number. The length of the bar can be any integer value with a minimum of 1 and a maximum of 255 character spaces, in practice the maximum line length will be 25 - the screen height. A variable allows the character used to draw the line to be defined in the Basic calling program. The variable locations required by the routine are as follows:

86 .... column number of vertical line start (top end).
87 .... screen line number of vertical line start.
88 .... length of vertical line.
00 .... ASCII code value of characters used in vertical line.

The routine is called from a Basic program with the command:

SYS(30052)

Some example applications for this function:
Demarcation between text or data areas, eg: columns of numbers.
To erase columns by using the ASCII space code in 00.
In drawing graphic diagrams.

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  : REM START AT COLUMN # 5
 20 POKE 87,10 : REM ON LINE # 10
 30 POKE 88,10 : REM LINE LENGTH 10 CHARACTERS
 40 POKE 0,160 : REM USE RVS SPACE CHARACTERS FOR LINE
 50 SYS(30052) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE            LINE

0337  7564                  ;**************************************
0338  7564                  ;* DRAW A VERTICAL BAR OF ANY PET
0339  7564                  ;* CHARACTER. CHARACTER STORED IN $00
0340  7564                  ;* COLUMN START IN $56, LINE OF START
0341  7564                  ;* IN $57, LENGTH OF LINE IN $58.
0342  7564                  ;* ROUTINE CALLED BY - SYS(30052)
0343  7564                  ;**************************************
0344  7564                  ;
0345  7564                  ;
0346  7564  48             VBAR   PHA
0347  7565  98                    TYA
0348  7566  48                    PHA
0349  7567  8A                    TXA
0350  7568  48                    PHA
0351  7569  20 00 74              JSR ADR1
0352  756C  A0 00                 LDY #$00
0353  756E  A6 58                 LDX T5
0354  7570  A5 00       VBAR2     LDA T8
0355  7572  91 54                 STA (T1),Y
0356  7574  18                    CLC
0357  7575  A5 54                 LDA T1
0358  7577  69 28                 ADC #$28
0359  7579  85 54                 STA T1
0360  757B  A5 55                 LDA T2
0361  757D  69 00                 ADC #$00
0362  757F  85 55                 STA T2
0363  7581  CA                    DEX
0364  7582  D0 EC                 BNE VBAR2
0365  7584  68                    PLA
0366  7585  AA                    TAX
0367  7586  68                    PLA
0368  7587  A8                    TAY
0369  7588  68                    PLA
0370  7589  60                    RTS
0371  758A                  ;
```

Bytes = 38

Fill Screen Block With Specified Character.

BLOCK is a routine to fill a designated block of the screen with a previously defined character. The position of the top left corner of the block is determined by two variables - column number and line number. The width of the block can be any integer value with a minimum of 1 and a maximum of 255 character spaces. The height of the block can be any integer value with a minimum of 1 and a maximum of 255, in practice of course the maximum block size will be the whole screen. A variable allows the character used to fill the block to be defined in the Basic calling program. The variable locations required by the routine are as follows:

86 .... column number of block start (top left corner).
87 .... screen line number of block start (top left corner).
88 .... width of block.
89 .... height of block.
00 .... ASCII code value of character used in block.

The routine is called from a Basic program with the command:

SYS(30470)

Some example applications for this function:
To give background characters in graphics.
To highlight data displayed in block area.
If the character used is a space then this function can be used to erase blocks of the screen.

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM START AT COLUMN # 10
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,8  : REM BLOCK WIDTH 8 CHARACTERS
40 POKE 89,5  : REM BLOCK HEIGHT 5 CHARACTERS
50 POKE 0,42  : REM USE STAR CHARACTERS FOR BLOCK
60 SYS(30470) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0644  7706                ;**************************************
0645  7706                ;* FILL A BLOCK OF THE SCREEN WITH A
0646  7706                ;* SPECIFIED CHARACTER. CHARACTER
0647  7706                ;* STORED IN $00, TOP LEFT COORDINATES
0648  7706                ;* IN $56 (COL) AND $57 (LINE), HEIGHT
0649  7706                ;* OF BLOCK IN $59, WIDTH IN $58
0650  7706                ;* ROUTINE CALLED BY - SYS(30470)
0651  7706                ;**************************************
0652  7706                ;
0653  7706                ;
0654  7706  48       BLOCK  PHA
0655  7707  98              TYA
0656  7708  48              PHA
0657  7709  8A              TXA
0658  770A  48              PHA
0659  770B  20 00 74        JSR ADR1
0660  770E  A6 59           LDX T6
0661  7710  A4 58    BLOCK1 LDY T5          ;FILL LINE WITH
0662  7712  88              DEY             ;CHARACTER
0663  7713  A5 00    BLOCK2 LDA T8
0664  7715  91 54           STA (T1),Y
0665  7717  88              DEY
0666  7718  10 F9           BPL BLOCK2
0667  771A  18              CLC
0668  771B  A5 54           LDA T1          ;CALCULATE START OF
0669  771D  69 28           ADC #$28        ;NEXT LINE
0670  771F  85 54           STA T1
0671  7721  90 02           BCC BLOCK3
0672  7723  E6 55           INC T2
0673  7725  CA       BLOCK3 DEX
0674  7726  D0 E8           BNE BLOCK1
0675  7728  68              PLA
0676  7729  AA              TAX
0677  772A  68              PLA
0678  772B  A8              TAY
0679  772C  68              PLA
0680  772D  60              RTS
0681  772E
```

Bytes: 40

Reverse Field Block of Screen.

INVT will reverse the field of all characters within a given block of screen area. The position of the top left of the block is determined by two variables - column number and line number. The width of the block can be any integer value with a minimum of 1 and a maximum of 255 character spaces. The height of the block can be any integer value with a minimum of 1 and a maximum of 255, though in practice the maximum size of the reverse field area is the whole screen. The variable locations required by the routine are as follows:

86 .... column number of reverse block start (top left corner).
87 .... screen line number of reverse block start.
88 .... width of reverse block area.
89 .... height of reverse block area

The routine is called from a Basic program with the command:

SYS(30510)

Some example applications for this function:

To highlight specific sets of data.

By repeatedly calling this function with a program loop on the same text area a flashing message can be generated, ideal for error and warning signs.

The following is an example of a Basic program using this routine:

```
 10 POKE 86,10 : REM START AT COLUMN # 10
 20 POKE 87,10 : REM ON LINE # 10
 30 POKE 88,8  : REM BLOCK WIDTH 8 CHARACTERS
 40 POKE 89,5  : REM BLOCK HEIGHT 5 CHARACTERS
 50 SYS(30510) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE                LINE

0683  772E                      ;**************************************
0684  772E                      ;* INVERT A BLOCK OF THE SCREEN
0685  772E                      ;* TOP LEFT COORDINATES IN $56 (COL)
0686  772E                      ;* AND $57 (LINE), HEIGHT OF BLOCK
0687  772E                      ;* IN $59, WIDTH IN $58
0688  772E                      ;* ROUTINE CALLED BY - SYS(30510)
0689  772E                      ;**************************************
0690  772E                      ;
0691  772E                      ;
0692  772E  48           INVT   PHA
0693  772F  98                  TYA
0694  7730  48                  PHA
0695  7731  8A                  TXA
0696  7732  48                  PHA
0697  7733  20 00 74            JSR ADR1
0698  7736  A6 59               LDX T6
0699  7738  A4 58         INVT1 LDY T5
0700  773A  88                  DEY
0701  773B  B1 54         INVT2 LDA (T1),Y      ;READ SCREEN CHARACTER
0702  773D  49 80               EOR #$80        ;INVERT IT
0703  773F  91 54               STA (T1),Y      ;AND REPLACE ON SCREEN
0704  7741  88                  DEY
0705  7742  10 F7               BPL INVT2
0706  7744  18                  CLC
0707  7745  A5 54               LDA T1
0708  7747  69 28               ADC #$28
0709  7749  85 54               STA T1
0710  774B  90 02               BCC INVT3
0711  774D  E6 55               INC T2
0712  774F  CA           INVT3  DEX
0713  7750  D0 E6               BNE INVT1
0714  7752  68                  PLA
0715  7753  AA                  TAX
0716  7754  68                  PLA
0717  7755  A8                  TAY
0718  7756  68                  PLA
0719  7757  60                  RTS
0720  7758                ;
```

Bytes = 42 -

Draw a Border Around Screen Block.

BORD will draw a thin line border around a specified area of he screen. The position of the top left corner of the border is determined by two variables - column number and line number. The width of the border can be any integer value with a minimum of 1 and a maximum of 255 character spaces. The height of the border can be any integer value with a minimum of 1 and a maximum of 255, in practice the maximum size of the border is around the outside edge of the screen. The variable locations required by the routine are as follows:

86 .... column number of border start (top left corner).
87 .... screen line number of border start.
88 .... width of border.
89 .... height of border.

The routine is called from a Basic program with the command:

SYS(30090)

Some example applications for this function:

Enhance the screen apperance by putting a border around the screen edge.

To identify associated data items by dividing the screen into sub screen areas using borders to isolate each area.

To isolate graphical displays from text etc.

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM TOP LEFT CORNER AT COLUMN # 10
20 POKE 87,5  : REM ON LINE # 5
30 POKE 88,20 : REM BORDER WIDTH 20 CHARACTERS
40 POKE 89,15 : REM BORDER HEIGHT 15 CHARACTERS
50 SYS(30090) : REM CALL SUBROUTINE
READY.
```

Actual Bytes used
for 'Address' + all up to here
= 389. 394

But Actual memory involved
through spacing Sub-routines
= 856.

```
LINE# LOC    CODE            LINE

0373  758A                  ;**************************************
0374  758A                  ;* DRAW A BORDER OF ANY SIZE AND ANY
0375  758A                  ;* LOCATION. TOP LEFT COORDINATES ARE
0376  758A                  ;* COLUMN IN $56, AND LINE IN $57
0377  758A                  ;* WIDTH IN $58 AND HEIGHT IN $59
0378  758A                  ;* ROUTINE CALLED BY - SYS(30090)
0379  758A                  ;**************************************
0380  758A                  ;
0381  758A                  ;
0382  758A  48             BORD   PHA
0383  758B  98                    TYA
0384  758C  48                    PHA
0385  758D  8A                    TXA
0386  758E  48                    PHA
0387  758F  20 00 74              JSR ADR1          ;TOP LEFT ADDRESS
0388  7592  A5 54                LDA T1            ;SAVE TOP LEFT ADDRESS
0389  7594  85 56                STA T3            ;IN $56 AND $57
0390  7596  A5 55                LDA T2
0391  7598  85 57                STA T4
0392  759A  A9 64                LDA #$64
0393  759C  85 00                STA T8
0394  759E  20 FA 75              JSR HBORD         ;DRAW TOP HORIZONTAL
0395  75A1  18                   CLC
0396  75A2  A5 54                LDA T1            ;FOR VERTICAL LINES
0397  75A4  69 29                ADC #$29          ;TOP ADDRESS = TOP
0398  75A6  85 54                STA T1            ;ADDRESS + 40.
0399  75A8  85 56                STA T3
0400  75AA  A5 55                LDA T2
0401  75AC  69 00                ADC #$00
0402  75AE  85 55                STA T2
0403  75B0  85 57                STA T4
0404  75B2  A9 65                LDA #$65
0405  75B4  85 00                STA T8
0406  75B6  20 04 76              JSR VBORD         ;DRAW LEFT VERTICAL
0407  75B9  18                   CLC
0408  75BA  A5 56                LDA T3            ;TOP VERTICAL ADDRESS
0409  75BC  65 58                ADC T5            ;= TOP VERTICAL + WIDTH
0410  75BE  85 54                STA T1
0411  75C0  C6 54                DEC T1
0412  75C2  A5 57                LDA T4
0413  75C4  69 00                ADC #$00
0414  75C6  85 55                STA T2
0415  75C8  A9 67                LDA #$67
0416  75CA  85 00                STA T8
0417  75CC  20 04 76              JSR VBORD         ;DRAW RIGHT VERTICAL
0418  75CF  A5 56                LDA T3
0419  75D1  85 54                STA T1
0420  75D3  A5 57                LDA T4
0421  75D5  85 55                STA T2
0422  75D7  A6 59                LDX T6            ;BOTTOM LEFT ADDRESS
0423  75D9  C6 54                DEC T1
0424  75DB  CA                   DEX
0425  75DC  CA                   DEX
0426  75DD  18                   CLC               ;CALCULATED
0427  75DE  A5 54                LDA T1
```

3AE8 3AF0 3AF8 3B00 3B08 3B10 3B18 3B20 3B28 3B30

3B39

00 74 = 3910 = DO 39

FA 75 = 3B50 = 50 3B

04 76 = 5A 3B

FA 75 = 50 3B

```
LINE# LOC    CODE          LINE

0428  75E0   69 28         ADC #$28        ;ADD 40 TO ADDRESS
0429  75E2   85 54         STA T1          ;EQUAL TO BOTTOM LEFT
0430  75E4   A5 55         LDA T2
0431  75E6   69 00         ADC #$00
0432  75E8   85 55         STA T2
0433  75EA   CA            DEX             ;DO AGAIN UNTIL HEIGHT
0434  75EB   D0 F0         BNE BORD1       ;IN X REG =0
0435  75ED   A9 63         LDA #$63
0436  75EF   85 00         STA T8
0437  75F1   20 FA 75      JSR HBORD       ;DRAW BOTTOM HORIZONTAL
0438  75F4   68            PLA
0439  75F5   AA            TAX
0440  75F6   68            PLA
0441  75F7   A8            TAY
0442  75F8   68            PLA
0443  75F9   60            RTS
0444  75FA   A4 58      HBORD  LDY T5          ;LINE LENGTH
0445  75FC   A5 00      HBORD1 LDA T8          ;CHARACTER FOR LINE
0446  75FE   91 54            STA (T1),Y      ;DISPLAY AT ADDRESS,Y
0447  7600   88            DEY
0448  7601   D0 F9         BNE HBORD1      ;END OF LINE?
0449  7603   60            RTS
0450  7604   A0 00      YBORD  LDY #$00
0451  7606   A6 59            LDX T6
0452  7608   CA               DEX
0453  7609   CA               DEX
0454  760A   A5 00      YBORD1 LDA T8
0455  760C   91 54            STA (T1),Y
0456  760E   18               CLC
0457  760F   A5 54            LDA T1          ;LOCATION FOR NEXT
0458  7611   69 28            ADC #$28        ;CHARACTER
0459  7613   85 54            STA T1
0460  7615   A5 55            LDA T2
0461  7617   69 00            ADC #$00
0462  7619   85 55            STA T2
0463  761B   CA               DEX
0464  761C   D0 EC         BNE VBORD1      ;END OF LINE?
0465  761E   60            RTS
0466  761F
```

3B38 3B40 3B48 3B50 3B58 3B5A 3B60 3B68 3B70

Bytes = 149

EXAMPLE 1

```
10 REM  ************************************
20 REM  *DEMONSTRATION PROGRAM TO SHOW SOME OF THE APPLICATIONS FOR
30 REM  *THE FOLLOWING ROUTINES IN THE GRAPHICS PACKAGE:
40 REM  * CURSOR CONTROL - SYS(31243)
50 REM  * HORIZONTAL LINE - SYS(30029)
60 REM  * VERTICAL LINE - SYS(30052)
70 REM  * BORDERS - SYS(30090)
80 REM  * BLOCK CHARACTER DISPLAY - SYS(30470)
90 REM  * BLOCK REVERSE FIELD - SYS(30510)
100 REM *THIS PROGRAM IS JUST PART OF A HYPOTHETICAL APPLICATION
110 REM *AND IS ONLY INTENDED AS A DEMONSTRATION.
120 REM
130 POKE 48,0:POKE49,112
140 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
150 T=0:GT=0:L=4:LA=18
160 PRINT"□"
170 REM
180 REM
190 REM * SET UP BORDERS.
200 REM
210 REM
220 POKE 86,0:POKE87,0:POKE88,38:POKE89,15:SYS(30090)
230 POKE 86,0:POKE87,15:POKE88,18:POKE89,10:SYS(30090)
240 POKE 86,20:POKE87,15:POKE88,18:POKE89,10:SYS(30090)
250 REM
260 REM
270 REM * SET UP HORIZONTAL AND VERTICAL BARS
280 REM
290 REM
300 POKE 86,20:POKE87,2:POKE88,9:POKE0,84:SYS(30052)
310 POKE 86,25:POKE87,2:POKE88,9:POKE0,84:SYS(30052)
320 POKE 86,31:POKE87,2:POKE88,9:POKE0,84:SYS(30052)
330 POKE 86,30:POKE87,11:POKE88,7:POKE0,70:SYS(30029)
340 REM
350 REM
360 REM * DISPLAY CHARACTER BLOCK TO INDICATE
370 REM * ADDRESS INPUT LIMITS.
380 REM
390 REM
400 POKE 86,2:POKE87,18:POKE88,16:POKE89,5:POKE0,100:SYS(30470)
410 REM
420 REM
430 REM * SET UP SCREEN TEXT USING CURSOR CONTROL ROUTINE
440 REM * EACH ITEM OF TEXT IS HELD AS THREE ELEMENTS IN A DATA
450 REM * STATEMENT - THEY ARE COLUMN #, LINE #, AND TEXT
460 REM
470 REM
480 DATA 4,2,DESCRIPTION
490 DATA 21,2,QTY
500 DATA 26,2,COST
510 DATA 32,2,TOTAL
520 DATA 25,12,TOTAL
```

```
530 DATA 3,16,NAME & ADDRESS
540 DATA 23,16,FUNCTION - ?
550 DATA 23,18,1 - UPDATE
560 DATA 23,20,2 - DELETE
570 DATA 23,22,3 - EXIT
580 FORQ=1TO10
590 READA:POKE85,A: REM * COLUMN #
600 READA:POKE84,A: REM * LINE #
610 SYS(31243):     REM * MOVE CURSOR
620 READA$:PRINTA$: REM * DISPLAY TEXT
630 NEXTQ
640 REM
650 REM
660 REM * SELECT AND RVS FIELD ACTIVE SCREEN FUNCTION FROM MENU
670 REM * ENCLOSED IN BOTTOM LEFT BORDER
680 REM
690 REM
700 REM
710 REM * RVS TO HIGHLIGHT WORD 'FUNCTION' IN MENU
720 REM
730 POKE 86,23:POKE87,16:POKE88,12:POKE89,1:SYS(30510)
740 GETA$:IFA$=""THEN740
750 REM
760 REM * RVS TO REMOVE HIGHLIGHT FROM 'FUNCTION'
770 REM
780 POKE 86,23:POKE87,16:POKE88,12:POKE89,1:SYS(30510)
790 A=VAL(A$):ONA GOTO830,2300,2630
800 GOTO640
810 REM
820 REM
830 REM * UPDATE FUNCTION DEMO ONLY
840 REM
850 REM
860 REM
870 REM * RVS TO HIGHLIGHT 'UPDATE' IN MENU
880 REM
890 POKE 86,23:POKE87,18:POKE88,10:POKE89,1:SYS(30510)
900 IFL<10THEN 1040
910 REM
920 REM * CURSOR CONTROL TO DISPLAY ERROR MESSAGE
930 REM
940 POKE84,12:POKE85,2:SYS(31243)
950 PRINT"■RECORD FULL■"
960 GOTO1880
970 REM
980 REM * INPUT DESCRIPTION
990 REM
1000 REM
1010 REM * HIGHLIGHT 'DESCRIPTION AND PUT CURSOR TO
1020 REM * DISPLAY INPUT ON LINE 'L'.
1030 REM
1040 POKE 86,4:POKE87,2:POKE88,11:POKE89,1:SYS(30510)
1050 POKE84,L:POKE85,4:SYS(31243)
1060 D$=""
1070 GETA$:IFA$=""THEN1070
1080 REM
1090 REM * IF INPUT IS '↑' THEN ABORT TO NEXT SECTION AND
1100 REM * REMOVE HIGHLIGHT FROM 'DESCRIPTION'
1110 REM
1120 IFA$="↑"THENPOKE 86,4:POKE87,2:POKE88,11:POKE89,1:SYS(30510):GOTO2000
```

```
1130 D$=D$+A$:IFA$=CHR$(13)THENGOTO1180
1140 PRINTA$;: GOTO1070
1150 REM
1160 REM * REMOVE HIGHLIGHT RVS FROM 'DESCRIPTION'
1170 REM
1180 POKE 86,4:POKE87,2:POKE88,11:POKE89,1:SYS(30510)
1190 REM
1200 REM * INPUT QUANTITY
1210 REM
1220 REM * RVS FIELD TO HIGHLIGHT 'QTY' AND MOVE CURSOR TO
1230 REM * DISPLAY INPUT ON LINE 'L'.
1240 REM
1250 POKE 86,21:POKE87,2:POKE88,3:POKE89,1:SYS(30510)
1260 POKE84,L:POKE85,21:SYS(31243)
1270 Q$=""
1280 GETA$:IFA$=""THEN1280
1290 Q$=Q$+A$:IFA$=CHR$(13)THENGOTO1340
1300 PRINTA$;: GOTO1280
1310 REM
1320 REM * REMOVE RVS HIGHLIGHT FROM 'QTY'
1330 REM
1340 POKE 86,21:POKE87,2:POKE88,3:POKE89,1:SYS(30510)
1350 REM
1360 REM * INPUT COST
1370 REM
1380 REM * RVS TO HIGHLIGHT 'COST' AND PUT CURSOR ON LINE 'L'
1390 REM * TO DISPLAY INPUT
1400 REM
1410 POKE 86,26:POKE87,2:POKE88,4:POKE89,1:SYS(30510)
1420 POKE84,L:POKE85,26:SYS(31243)
1430 C$=""
1440 GETA$:IFA$=""THEN1440
1450 C$=C$+A$:IFA$=CHR$(13)THENGOTO1500
1460 PRINTA$;: GOTO1440
1470 REM
1480 REM * REMOVE RVS HIGHLIGHT FROM 'COST'
1490 REM
1500 POKE 86,26:POKE87,2:POKE88,4:POKE89,1:SYS(30510)
1510 REM
1520 REM * CALCULATE AND DISPLAY TOTAL
1530 REM
1540 REM * RVS TO HIGHLIGHT 'TOTAL'
1550 REM
1560 POKE 86,32:POKE87,2:POKE88,5:POKE89,1:SYS(30510)
1570 T=VAL(Q$)*VAL(C$):GT=GT+T
1580 REM
1590 REM * PUT CURSOR ON LINE 'L' AT POSITION TO DISPLAY TOTAL
1600 REM
1610 POKE84,L:POKE85,32:SYS(31243)
1620 PRINTT
1630 REM
1640 REM * REMOVE RVS HIGHLIGHT FROM 'TOTAL'
1650 REM
1660 POKE 86,32:POKE87,2:POKE88,5:POKE89,1:SYS(30510)
1670 REM
1680 REM * PROMPT FOR ANOTHER ENTRY
1690 REM
1700 REM * MOVE CURSOR TO LOCATION FOR PROMPT AND DISPLAY
1710 REM
1720 POKE 84,13:POKE85,2:SYS(31243)
```

```
1730 PRINT"■ANOTHER ENTRY - Y OR N■"
1740 GETA$:IFA$=""THEN1740
1750 L=L+1
1760 REM
1770 REM * USE HORIZONTAL BAR OF SPACE CHARS TO DELETE PROMPT
1780 REM
1790 POKE86,1:POKE87,13:POKE88,22:POKE0,32:SYS(30029)
1800 IFA$="N"THEN1880
1810 GOTO900
1820 REM
1830 REM * CALCULATE GRAND TOTAL FOR ALL ENTRIES
1840 REM
1850 REM * RVS TO HIGHLIGHT 'TOTAL' AND MOVE CURSOR TO
1860 REM * LOCATION TO DISPLAY GRAND TOTAL
1870 REM
1880 POKE86,25:POKE87,12:POKE88,5:POKE89,1:SYS(30510)
1890 POKE84,12:POKE85,32:SYS(31243)
1900 PRINTGT:FORQ=1TO1000:NEXTQ
1910 REM
1920 REM * REMOVE RVS HIGHLIGHT FROM 'TOTAL'
1930 REM
1940 POKE86,25:POKE87,12:POKE88,5:POKE89,1:SYS(30510)
1950 REM
1960 REM * INPUT NAME AND ADDRESS
1970 REM
1980 REM * RVS HIGHLIGHT 'NAME & ADDRESS'
1990 REM
2000 POKE 86,3:POKE87,16:POKE88,14:POKE89,1:SYS(30510)
2010 LA=18
2020 REM
2030 REM * MOVE CURSOR TO START OF NAME AND ADDRESS INPUT
2040 REM * ON LINE 'LA'
2050 REM
2060 POKE84,LA:POKE85,2:SYS(31243)
2070 GETA$:IFA$=""THEN2070
2080 IFA$="↑"THEN2210
2090 IFA$=CHR$(13)THEN2070
2100 PRINTA$;:AD$=AD$+A$
2110 IFA$=","THENLA=LA+1:GOTO2140
2120 IFA$="."THENGOTO2160
2130 GOTO2070
2140 IFLA>22 THEN LA=22
2150 GOTO2060
2160 GETA$:IFA$=""THEN2160
2170 IFA$<>CHR$(13)THEN2070
2180 REM
2190 REM * RVS TO REMOVE HIGHLIGHT FROM 'NAME & ADDRESS'
2200 REM
2210 POKE 86,3:POKE87,16:POKE88,14:POKE89,1:SYS(30510)
2220 REM
2230 REM * REMOVE HIGHLIGHT FROM FUNCTION 'UPDATE'
2240 REM * AND RETURN TO FUNCTION SELECT
2250 REM
2260 POKE 86,23:POKE87,18:POKE88,10:POKE89,1:SYS(30510)
2270 GOTO640
2280 REM
2290 REM
2300 REM * DELETE RECORD FUNCTION
2310 REM
2320 REM
```

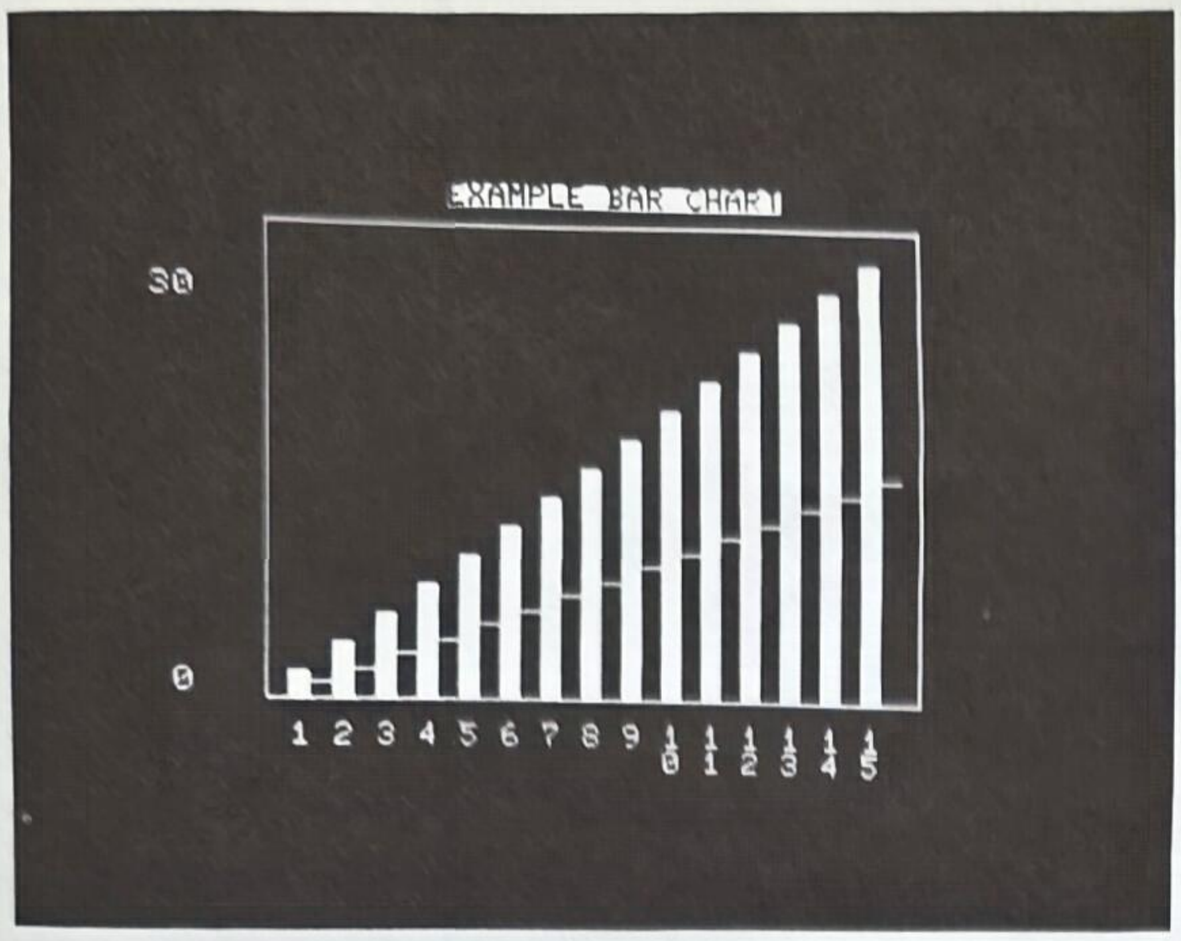
```
2330 D$="":Q$="":C$="":AD$=""
2340 REM
2350 REM * RVS TO HIGHLIGHT 'DELETE' FUNCTION
2360 REM
2370 POKE 86,23:POKE87,20:POKE88,10:POKE89,1:SYS(30510)
2380 T=0:GT=0:L=4:LA=18
2390 REM
2400 REM * DELETE BLOCKS OF TEXT FROM SCREEN USING THE ROUTINE
2410 REM * TO FILL A SCREEN BLOCK WITH SPECIFIED CHARACTER
2420 REM * - HERE A SPACE CHARACTER.
2430 REM
2440 POKE86,2:POKE87,4:POKE88,18:POKE89,7:POKE0,32:SYS(30470)
2450 POKE86,21:POKE87,4:POKE88,4:POKE89,7:POKE0,32:SYS(30470)
2460 POKE86,26:POKE87,4:POKE88,4:POKE89,7:POKE0,32:SYS(30470)
2470 POKE86,32:POKE87,4:POKE88,6:POKE89,7:POKE0,32:SYS(30470)
2480 REM
2490 REM * DELETE LINE USING HORIZONTAL LINE DRAWING WITH
2500 REM * SPACE CHARACTER.
2510 REM
2520 POKE 86,32:POKE87,12:POKE88,5:POKE0,32:SYS(30029)
2530 REM
2540 REM * DISPLAY CHARACTER BLOCK TO GIVE ADDRESS INPUT LIMITS
2550 REM
2560 POKE 86,2:POKE87,18:POKE88,16:POKE89,5:POKE0,100:SYS(30470)
2570 REM
2580 REM * RVS TO REMOVE HIGHLIGHT FROM 'DELETE' AND
2590 REM * RETURN TO FUNCTION SELECT.
2600 REM
2610 POKE 86,23:POKE87,20:POKE88,10:POKE89,1:SYS(30510)
2620 GOTO640
2630 STOP
READY.
```

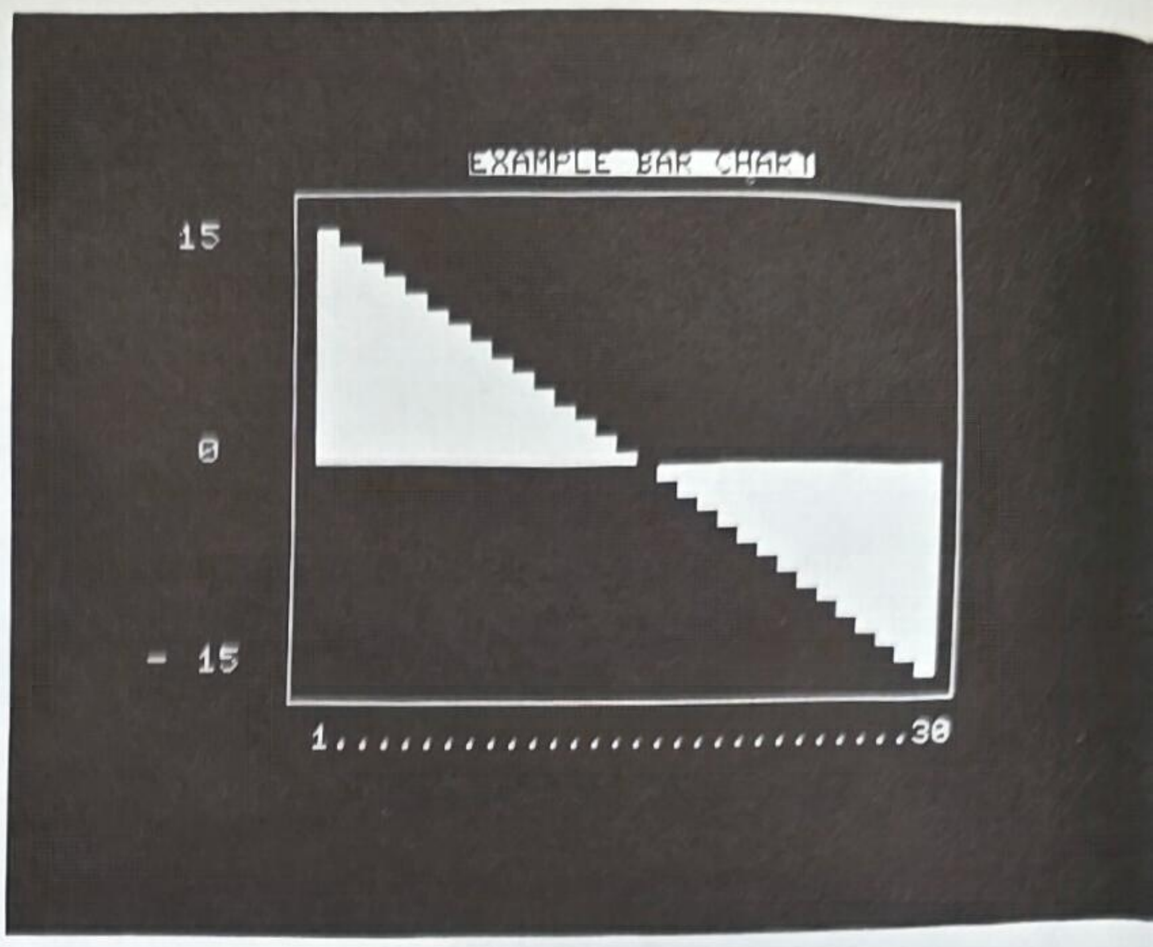
Notes

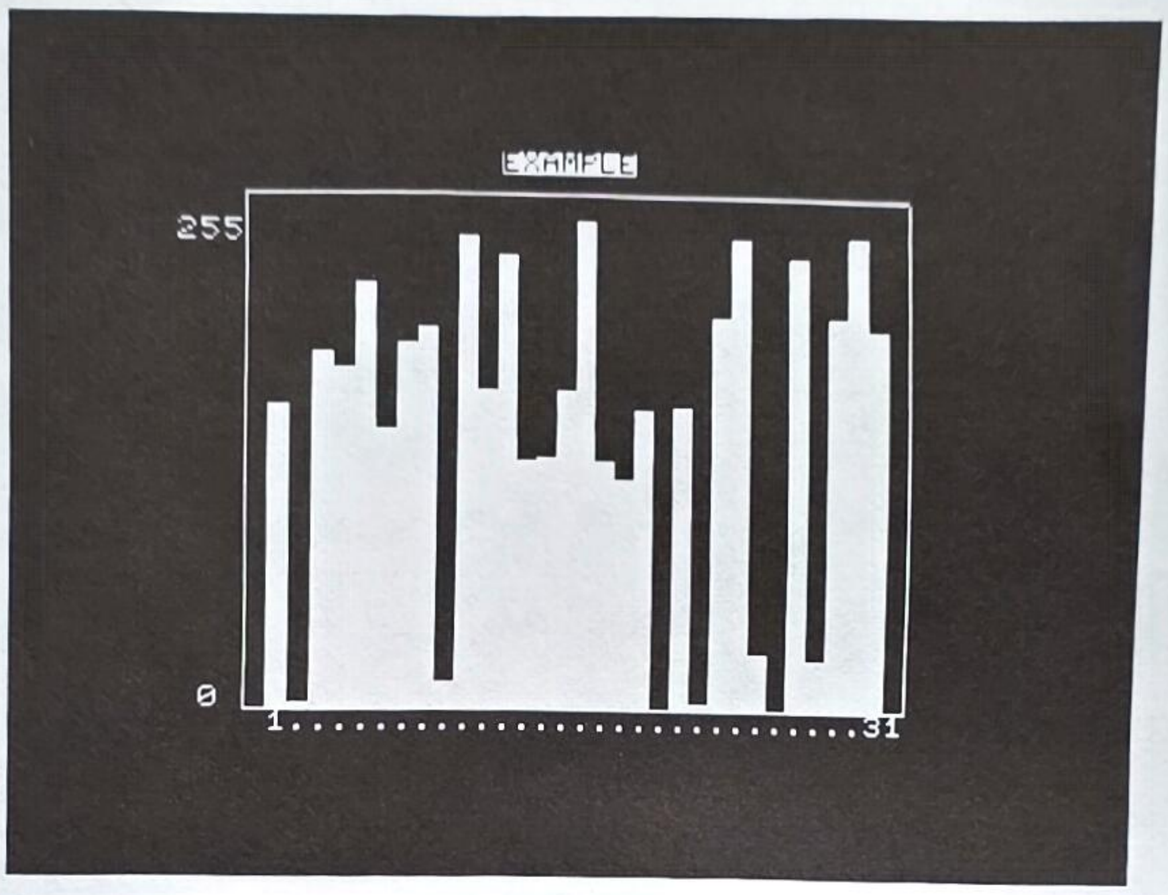
3 FINE RESOLUTION PLOTTING

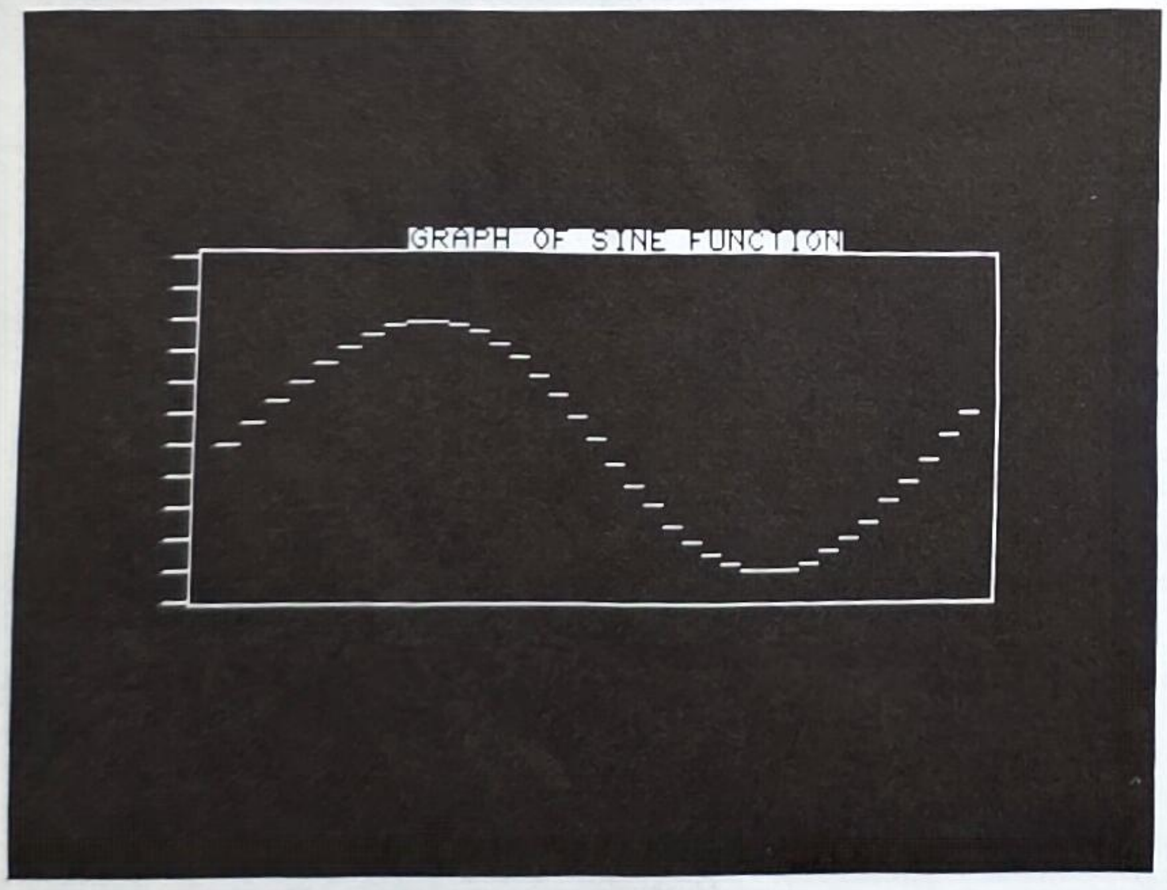
Section 3

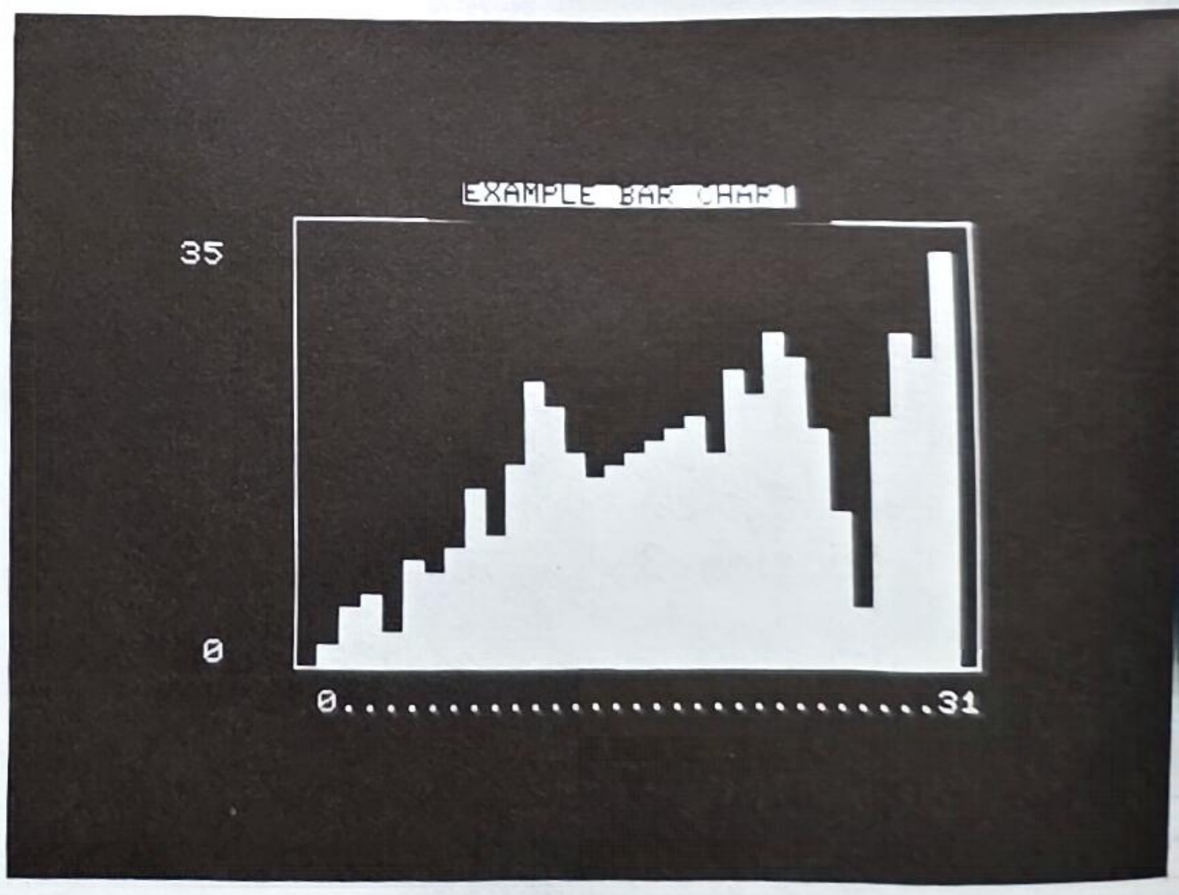
Fine resolution bar plotting with machine code - 65
Plot fine resolution bar, horizontal, left to right - 68
Plot fine resolution bar, horizontal, right to left - 70
Plot fine resolution bar, vertical, bottom to top - 72
Plot fine resolution bar, vertical, top to bottom - 74
Plot fine resolution line, horizontal, left to right - 76
Plot fine resolution line, horizontal, right to left - 78
Plot fine resolution line, vertical, bottom to top - 80
Plot fine resolution line, vertical, top to bottom - 82
Example application emulates analog edge meter - 84
Example application draws the graph of a function - 85
Example draws a barchart of 31 +ve values - 86
Example draws a barchart, 2 data sets of 15 +ve values - 87
Example draws a barchchart of 30 +ve or -ve values - 88

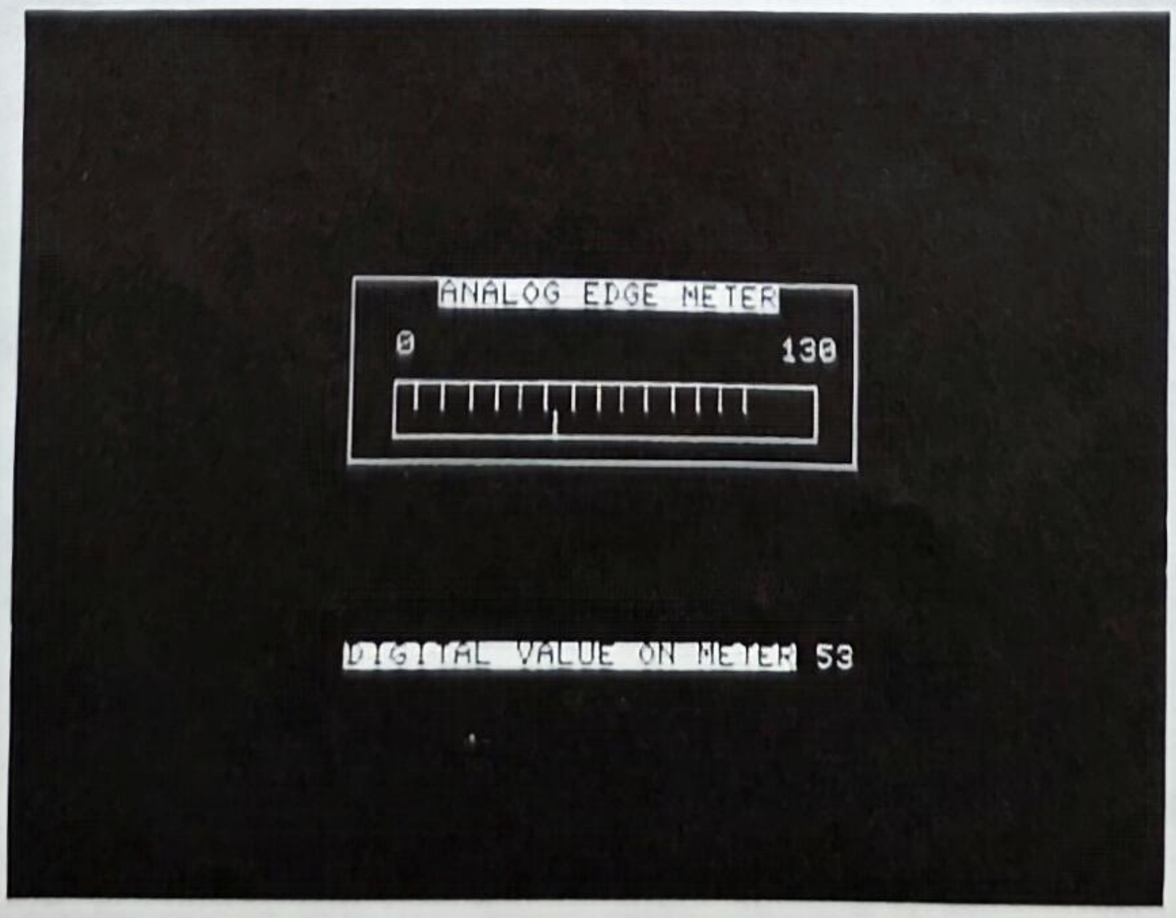

EXAMPLE BAR CHART
30
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


EXAMPLE BAR CHART
15
0
- 15
1.....30


EXAMPLE
255
0
1.....31


GRAPH OF SINE FUNCTION


EXAMPLE BAR CHART
35
0
0.....31


ANALOG EDGE METER
0
130
DIGITAL VALUE ON METER 53

FINE RESOLUTION PLOTTING 3

A great drawback in having a display only 40 or 80 characters wide and 25 lines deep is the poor definition achievable when displaying data in graphical form. A limitation particularly apparent when trying to display data as bar charts, either vertical or horizontal, on the screen. Fortunately the graphics character set on the PET allows this limitation to be overcome. By careful use of specific graphic characters in both normal and reverse field a single character space can be resolved into eight lines in either horizontal or vertical directions. The resolution of a character space thus acheved can take two forms, either a set of eight characters each having a reverse field line of increasing width (from 1/8 character to 8/8 of a character), or a set of characters each with a single thin line but positioned in the 1/8 character increments.

The first of these forms can be used with normal reverse field spaces to create bar charts with resolutions on a 40 column machine of one part in 320 in the horizontal and 200 in the vertical. The second form is used where a thin finely incremented line is required, such as the two example programs in this section, plotting a graph of a curve and the simulation of a moving coil edge meter. For each of these two forms of fine resolution plotting there are two sets of characters, one set for vertical and the other for horizontal plotting.

The way these sets of fine resolution plotting characters are used to display a bar or line in fine resolution is best demonstrated by examining the Basic program in Example 2. If the length of the bar is greater than eight then eight is subtracted from the length and a character - either a reverse or normal space - is displayed in the correct position at the end of the bar. This is repeated until the bar length remaining to be displayed is less than eight. The remaining length is then used as a pointer to one of the eight fine resolution characters in a character table, this character is then displayed at the end of the existing bar display (lines 24168 to 24184). The machine code routines employ the same method as that used in the Basic program.

The Basic program in Example 2 is designed as a standard subroutine for drawing fine resolution vertical barcharts and will plot up to 31 variables with a definition of 1 in 160. The variables are transfered to the subroutine as an integer array P%(X). The barchart is surrounded by a border and given a heading in reverse field characters in the centre of the top line of the screen. A vertical scale is given from zero to the maximum value to be displayed, the line increment is thus 1/160 of the maximum value stored in P%(X). The

horizontal scale numbers the bars from 1 to 31; this number wAs chosen to allow the bar chart to display daily data over a one month period and can be changed if desired. This subroutine uses the following parameters:

P$ - variable for table heading; maximum length 40 characters.
P%(X) - table of data to be displayed in barchart; maximum 31 entries, all integers.
SP$ - string variable of 40 space characters.

EXAMPLE 2

```
100 DIMP%(31)
110 SP$="                                        "
1000 REM ************************************
1001 REM *EXAMPLE OF SUBROUTINE CALL
1002 REM *TO DRAW A BARCHART.
1009 REM ************************************
1010 FORQ=1TO31
1020 INPUTP:IFP<0THEN 1050:REM **INPUT TILL NEGATIVE NUMBER**
1030 P%(Q)=P
1040 NEXTQ
1050 P$="EXAMPLE":REM **HEADING**
1060 GOSUB24100
1070 GETA$:IFA$=""THEN 1070:REM **WAIT TILL KEY PRESSED TO END**
1080 END
2000 REM
3000 REM
4000 REM
5000 REM
6000 REM
24100 REM ************************************
24102 REM *SUBROUTINE TO DRAW BARCHART
24104 REM *USING 31 VARIABLES STORED
24106 REM *AS P%(X). TABLE HEADING IS
24108 REM *TRANSFERED AS P$.
24110 REM ************************************
24112 A=LEN(P$):A1=(40-A)/2
24114 A1$=LEFT$(SP$,A1)
24116 PRINT"■";
24118 PRINTA1$;:PRINT"■"P$"■"
24120 PRINT"    ■■■
```

```
24136 IFA>BTHENB=A
24138 NEXTQ
24140 PRINT"■■■■"B
24142 A=B/160
24144 FORQ=1TO31
24146 PRINT"■"
24148 FORAV=1TO21:PRINT"■";:NEXTAV
24150 PRINTTAB(Q+4);
24152 AS=INT(P%(Q)/A)
24154 AL=INT(AS/8)
24156 AF=AS-(8*AL)
24158 IFAL<1THENGOTO24166
24160 FORQS=0TOAL-1
24162 PRINT"■ ■";
24164 NEXTQS
24166 IFAF=0GOTO24186
24168 ONAFGOTO24170,24172,24174,24176,24178,24180,24182,24184
24170 PRINT"▌ ":GOTO24186
24172 PRINT"▌▏":GOTO24186
24174 PRINT"▌▎":GOTO24186
24176 PRINT"▌▍":GOTO24186
24178 PRINT"■▌":GOTO24186
24180 PRINT"■▋":GOTO24186
24182 PRINT"■▊":GOTO24186
24184 PRINT"■▉":GOTO24186
24186 NEXTQ
24188 RETURN
READY.
```

Fine Resolution Bar Plotting - Horizontal, Left to Right.

This routine draws a horizontal bar of reverse field spaces from left to right in fine resolution. The left hand coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution bar plot (left end).
87 .... line number of start of fine resolution plot.
88 .... length of fine resolution bar plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(29799)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  : REM START AT COLUMN # 5
 20 POKE 87,10 : REM ON LINE # 10
 30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
 40 SYS(29799) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE            LINE

0137  7467                  ;**************************************
0138  7467                  ;* DRAW HORIZONTAL BAR - LEFT TO RIGHT.
0139  7467                  ;* RESOLUTION  135 X 25. COLUMN # IN $56
0140  7467                  ;* LINE # IN $57, LENGTH IN $58
0141  7467                  ;* ROUTINE CALLED BY - SYS(29799)
0142  7467                  ;**************************************
0143  7467                  ;
0144  7467                  ;
0145  7467  48             HORIZ  PHA              ;SAVE REGISTERS
0146  7468  98                    TYA
0147  7469  48                    PHA
0148  746A  8A                    TXA
0149  746B  48                    PHA
0150  746C  20 00 74              JSR ADR1        ;CALCULATE ADDRESS
0151  746F  A0 00                 LDY #$00
0152  7471  A5 58           HOR2   LDA T5           ;LINE LENGTH < 8 ?
0153  7473  C9 08                 CMP #$08
0154  7475  30 12                 BMI HOR4        ;PLOT FRACTIONAL CHARACTER
0155  7477  E9 08                 SBC #$08
0156  7479  85 58                 STA T5          ;REDUCE LINE LENGTH BY 8
0157  747B  A9 A0                 LDA #$A0
0158  747D  91 54                 STA (T1),Y      ;AND DISPLAY RVS SPACE
0159  747F  18                    CLC             ;ON SCREEN
0160  7480  E6 54                 INC T1          ;INCREMENT ADDRESS
0161  7482  90 02                 BCC HOR3
0162  7484  E6 55                 INC T2
0163  7486  4C 71 74        HOR3   JMP HOR2         ;NEXT CHARACTER
0164  7489  AA              HOR4   TAX
0165  748A  BD 95 74              LDA HOR5,X      ;POINT TO FRACTIONAL
0166  748D  91 54                 STA (T1),Y      ;CHARACTER TABLE
0167  748F  68                    PLA             ;RESTORE REGISTERS
0168  7490  AA                    TAX
0169  7491  68                    PLA
0170  7492  A8                    TAY
0171  7493  68                    PLA
0172  7494  60                    RTS
0173  7495                  ;
0174  7495                  ;
0175  7495  20              HOR5   .BYTE $20,$65,$74,$75,$61
0175  7496  65
0175  7497  74
0175  7498  75
0175  7499  61
0176  749A  F6                    .BYTE $F6,$EA,$E7,$A0
0176  749B  EA
0176  749C  E7
0176  749D  A0
0177  749E                  ;
```

Bytes = 55

Fine Resolution Bar Plotting - Horizontal, Right to Left.

This routine draws a horizontal bar of reverse field spaces from right to left in fine resolution. The right hand coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous d splay. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution bar plot (right end).
87 .... line number of start of fine resolution plot.
88 .... length of fine resolution bar plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(29914)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,25 : REM START AT COLUMN #25
 20 POKE 87,10 : REM ON LINE # 10
 30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
 40 SYS(29914) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE              LINE

0223  74DA                    ;**************************************
0224  74DA                    ;* DRAW HORIZONTAL BAR - RIGHT TO LEFT.
0225  74DA                    ;* RESOLUTION  135 X 25. COLUMN # IN $56
0226  74DA                    ;* LINE # IN $57, LENGTH IN $58
0227  74DA                    ;* ROUTINE CALLED BY - SYS(29914)
0228  74DA                    ;**************************************
0229  74DA                    ;
0230  74DA                    ;
0231  74DA  48               RHOR1  PHA              ;SAVE REGISTERS
0232  74DB  98                      TYA
0233  74DC  48                      PHA
0234  74DD  8A                      TXA
0235  74DE  48                      PHA
0236  74DF  20 00 74                JSR ADR1        ;CALCULATE ADDRESS
0237  74E2  A0 00                   LDY #$00
0238  74E4  A5 58            RHOR2  LDA T5           ;LINE LENGTH < 8 ?
0239  74E6  C9 08                   CMP #$08
0240  74E8  30 12                   BMI RHOR4       ;PLOT FRACTIONAL CHARACTER
0241  74EA  E9 08                   SBC #$08
0242  74EC  85 58                   STA T5          ;REDUCE LINE LENGTH BY 8
0243  74EE  A9 A0                   LDA #$A0        ;AND DISPLAY RVS SPACE
0244  74F0  91 54                   STA (T1),Y      ;ON SCREEN
0245  74F2  38                      SEC
0246  74F3  C6 54                   DEC T1          ;DECREMENT ADDRESS
0247  74F5  B0 02                   BCS RHOR3
0248  74F7  C6 55                   DEC T2
0249  74F9  4C E4 74        RHOR3  JMP RHOR2        ;NEXT CHARACTER
0250  74FC  AA               RHOR4  TAX
0251  74FD  BD 08 75                LDA RHOR5,X     ;POINT TO FRACTIONAL
0252  7500  91 54                   STA (T1),Y      ;CHARACTER TABLE
0253  7502  68                      PLA             ;RESTORE REGISTERS
0254  7503  AA                      TAX
0255  7504  68                      PLA
0256  7505  A8                      TAY
0257  7506  68                      PLA
0258  7507  60                      RTS
0259  7508                    ;
0260  7508                    ;
0261  7508  20               RHOR5  .BYTE $20,$67,$6A,$76,$E1
0261  7509  67
0261  750A  6A
0261  750B  76
0261  750C  E1
0262  750D  F5                      .BYTE $F5,$F4,$E5,$A0
0262  750E  F4
0262  750F  E5
0262  7510  A0
0263  7511
```

Bytes = 55

Fine Resolution Bar Plotting - Vertical, Bottom to Top.

This routine draws a vertical bar of reverse field spaces from bottom to top in fine resolution. The bottom coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution bar plot (bottom end).
87 .... line number of start of fine resolution plot.
88 .... length of fine resolution bar plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(29854)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,10 : REM START AT COLUMN #10
 20 POKE 87,10 : REM ON LINE # 10
 30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
 40 SYS(29854) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC   CODE          LINE

0179  749E               ;**************************************
0180  749E               ;* DRAW VERTICAL BAR - BOTTOM TO TOP.
0181  749E               ;* RESOLUTION 135 X 40. COLUMN # IN $56
0182  749E               ;* LINE # IN $57, LENGTH IN $58
0183  749E               ;* ROUTINE CALLED BY - SYS(29854)
0184  749E               ;**************************************
0185  749E               ;
0186  749E  48          VERT   PHA                  ;SAVE REGISTERS
0187  749F  98                 TYA
0188  74A0  48                 PHA
0189  74A1  8A                 TXA
0190  74A2  48                 PHA
0191  74A3  20 00 74           JSR ADR1
0192  74A6  A0 00              LDY #$00
0193  74A8  A5 58        VER1   LDA T5
0194  74AA  C9 08              CMP #$08            ;LENGTH > 8 ?
0195  74AC  30 17              BMI VER2
0196  74AE  E9 08              SBC #$08            ;REDUCE LENGTH BY 8 AND
0197  74B0  85 58              STA T5
0198  74B2  A9 A0              LDA #$A0            ;DRAW RVS SPACE
0199  74B4  91 54              STA (T1),Y
0200  74B6  18                 CLC
0201  74B7  A5 54              LDA T1              ;SUBTRACT 40 FROM ADDRESS
0202  74B9  E9 27              SBC #$27
0203  74BB  85 54              STA T1
0204  74BD  A5 55              LDA T2
0205  74BF  E9 00              SBC #$00
0206  74C1  85 55              STA T2
0207  74C3  D0 E3              BNE VER1
0208  74C5  AA           VER2   TAX
0209  74C6  BD D1 74           LDA VER3,X
0210  74C9  91 54              STA (T1),Y
0211  74CB  68                 PLA
0212  74CC  AA                 TAX
0213  74CD  68                 PLA
0214  74CE  A8                 TAY
0215  74CF  68                 PLA
0216  74D0  60                 RTS
0217  74D1               ;
0218  74D1               ;
0219  74D1  20           VER3   .BYTE $20,$64,$6F,$79,$62
0219  74D2  64
0219  74D3  6F
0219  74D4  79
0219  74D5  62
0220  74D6  F8                 .BYTE $F8,$F7,$E3,$A0
0220  74D7  F7
0220  74D8  E3
0220  74D9  A0
0221  74DA               ;
```

Bytes = 60 .

Fine Resolution Bar Plotting - Vertical, Top to Bottom.

This routine draws a vertical bar of reverse field spaces from top to bottom in fine resolution. The left hand coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution bar plot (top end).
87 .... line number of start of fine resolution plot.
88 .... length of fine resolution bar plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(29969)

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM START AT COLUMN #10
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
40 SYS(29969) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0265  7511                ;**************************************
0266  7511                ;* DRAW VERTICAL BAR - TOP TO BOTTOM.
0267  7511                ;* RESOLUTION 135 X 40. COLUMN # IN $56
0268  7511                ;* LINE # IN $57, LENGTH IN $58
0269  7511                ;* ROUTINE CALLED BY - SYS(29969)
0270  7511                ;**************************************
0271  7511                ;
0272  7511  48           TVERT  PHA                  ;SAVE REGISTERS
0273  7512  98                  TYA
0274  7513  48                  PHA
0275  7514  8A                  TXA
0276  7515  48                  PHA
0277  7516  20 00 74            JSR ADR1
0278  7519  A0 00               LDY #$00
0279  751B  A5 58       TVER1  LDA T5
0280  751D  C9 08               CMP #$08           ;LENGTH > 8 ?
0281  751F  30 17               BMI TVER2
0282  7521  E9 08               SBC #$08           ;REDUCE LENGTH BY 8 AND
0283  7523  85 58               STA T5
0284  7525  A9 A0               LDA #$A0           ;DRAW RVS SPACE
0285  7527  91 54               STA (T1),Y
0286  7529  18                  CLC
0287  752A  A5 54               LDA T1             ;ADD 40 TO ADDRESS
0288  752C  69 28               ADC #$28
0289  752E  85 54               STA T1
0290  7530  A5 55               LDA T2
0291  7532  69 00               ADC #$00
0292  7534  85 55               STA T2
0293  7536  D0 E3               BNE TVER1
0294  7538  AA           TVER2  TAX
0295  7539  BD 44 75            LDA TVER3,X
0296  753C  91 54               STA (T1),Y
0297  753E  68                  PLA
0298  753F  AA                  TAX
0299  7540  68                  PLA
0300  7541  A8                  TAY
0301  7542  68                  PLA
0302  7543  60                  RTS
0303  7544                ;
0304  7544                ;
0305  7544  20           TVER3  .BYTE $20,$63,$77,$78,$E2
0305  7545  63
0305  7546  77
0305  7547  78
0305  7548  E2
0306  7549  F9                  .BYTE $F9,$EF,$E4,$A0
0306  754A  EF
0306  754B  E4
0306  754C  A0
0307  754D            ;
```

Bytes = 60

Fine Resolution Line Plotting - Horizontal, Left to Right.

This routine draws a horizontal bar of space characters from left to right terminated by a fine resolution vertical line one eighth of a character wide. The left hand coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution line plot (left end).
87 .... line number of start of fine resolution plot.
88 .... length of fine resolution line plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(30239)

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM START AT COLUMN #10
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
 40 SYS(30239) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0468  761F                ;**************************************
0469  761F                ;* DRAW HORIZONTAL BAR - LEFT TO RIGHT.
0470  761F                ;* OF SPACES TERMINATED IN A THIN LINE
0471  761F                ;* RESOLUTION  135 X 25. COLUMN # IN $56
0472  761F                ;* LINE # IN $57, LENGTH IN $58
0473  761F                ;* ROUTINE CALLED BY - SYS(30239)
0474  761F                ;**************************************
0475  761F                ;
0476  761F                ;
0477  761F  48     LHORIZ PHA                  ;SAVE REGISTERS
0478  7620  98            TYA
0479  7621  48            PHA
0480  7622  8A            TXA
0481  7623  48            PHA
0482  7624  20 00 74       JSR ADR1           ;CALCULATE ADDRESS
0483  7627  A0 00           LDY #$00
0484  7629  A5 58    LHOR2  LDA T5             ;LINE LENGTH < 8 ?
0485  762B  C9 07           CMP #$07
0486  762D  30 12           BMI LHOR4          ;PLOT FRACTIONAL CHARACTER
0487  762F  E9 07           SBC #$07
0488  7631  85 58           STA T5             ;REDUCE LINE LENGTH BY 8
0489  7633  A9 20           LDA #$20
0490  7635  91 54           STA (T1),Y         ;AND DISPLAY RVS SPACE
0491  7637  18              CLC                ;ON SCREEN
0492  7638  E6 54           INC T1             ;INCREMENT ADDRESS
0493  763A  90 02           BCC LHOR3
0494  763C  E6 55           INC T2
0495  763E  4C 29 76 LHOR3  JMP LHOR2          ;NEXT CHARACTER
0496  7641  AA      LHOR4  TAX
0497  7642  BD 4D 76         LDA LHOR5,X        ;POINT TO FRACTIONAL
0498  7645  91 54           STA (T1),Y         ;CHARACTER TABLE
0499  7647  68              PLA                ;RESTORE REGISTERS
0500  7648  AA              TAX
0501  7649  68              PLA
0502  764A  A8              TAY
0503  764B  68              PLA
0504  764C  60              RTS
0505  764D                ;
0506  764D                ;
0507  764D  65      LHOR5  .BYTE $65,$54,$47,$42
0507  764E  54
0507  764F  47
0507  7650  42
0508  7651  48              .BYTE $48,$59,$67,$20,$20
0508  7652  59
0508  7653  67
0508  7654  20
0508  7655  20
0509  7656                ;
```

Bytes = 55.

Fine Resolution Line Plotting - Horizontal, Right to Left.

This routine draws a horizontal bar of space characters from right to left terminated by a fine resolution vertical line one eighth of a character wide. The right hand coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution line plot (right end).

87 .... line number of start of fine resolution plot.

88 .... length of fine resolution line plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(30354)

The following is an example of a Basic program using this routine:

```
10 POKE 86,25 : REM START AT COLUMN #25
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
40 SYS(30354) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE            LINE

0556  7692                  ;**************************************
0557  7692                  ;* DRAW HORIZONTAL BAR - RIGHT TO LEFT.
0558  7692                  ;* OF SPACES TERMINATED IN A THIN LINE
0559  7692                  ;* RESOLUTION  135 X 25. COLUMN # IN $56
0560  7692                  ;* LINE # IN $57, LENGTH IN $58
0561  7692                  ;* ROUTINE CALLED BY - SYS(30354)
0562  7692                  ;**************************************
0563  7692                  ;
0564  7692                  ;
0565  7692  48        LRHOR1 PHA             ;SAVE REGISTERS
0566  7693  98               TYA
0567  7694  48               PHA
0568  7695  8A               TXA
0569  7696  48               PHA
0570  7697  20 00 74         JSR ADR1       ;CALCULATE ADDRESS
0571  769A  A0 00            LDY #$00
0572  769C  A5 58     LRHOR2 LDA T5          ;LINE LENGTH < 8 ?
0573  769E  C9 07            CMP #$07
0574  76A0  30 12            BMI LRHOR4     ;PLOT FRACTIONAL CHARACTER
0575  76A2  E9 07            SBC #$07
0576  76A4  85 58            STA T5         ;REDUCE LINE LENGTH BY 8
0577  76A6  A9 20            LDA #$20       ;AND DISPLAY RVS SPACE
0578  76A8  91 54            STA (T1),Y     ;ON SCREEN
0579  76AA  38               SEC
0580  76AB  C6 54            DEC T1         ;DECREMENT ADDRESS
0581  76AD  B0 02            BCS LRHOR3
0582  76AF  C6 55            DEC T2
0583  76B1  4C 9C 76  LRHOR3 JMP LRHOR2      ;NEXT CHARACTER
0584  76B4  AA        LRHOR4 TAX
0585  76B5  BD C0 76         LDA LRHOR5,X   ;POINT TO FRACTIONAL
0586  76B8  91 54            STA (T1),Y     ;CHARACTER TABLE
0587  76BA  68               PLA            ;RESTORE REGISTERS
0588  76BB  AA               TAX
0589  76BC  68               PLA
0590  76BD  A8               TAY
0591  76BE  68               PLA
0592  76BF  60               RTS
0593  76C0                  ;
0594  76C0                  ;
0595  76C0  67        LRHOR5 .BYTE $67,$59,$48,$42,$47
0595  76C1  59
0595  76C2  48
0595  76C3  42
0595  76C4  47
0596  76C5  54               .BYTE $54,$65,$20,$20
0596  76C6  65
0596  76C7  20
0596  76C8  20
0597  76C9                  ;
```

Bytes = 55.

Fine Resolution Line Plotting - Vertical, Bottom to Top.

This routine draws a vertical bar of space characters from bottom to top terminated by a fine resolution horizontal line one eighth of a character wide. The bottom coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution line plot (bottom end).

87 .... line number of start of fine resolution plot.

88 .... length of fine resolution line plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

SYS(30294)

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM START AT COLUMN #10
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
40 SYS(30294) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE            LINE

0511  7656                  ;**************************************
0512  7656                  ;* DRAW VERTICAL BAR - BOTTOM TO TOP.
0513  7656                  ;* OF SPACES TERMINATED IN A THIN LINE
0514  7656                  ;* RESOLUTION 135 X 40. COLUMN # IN $56
0515  7656                  ;* LINE # IN $57, LENGTH IN $58
0516  7656                  ;* ROUTINE CALLED BY - SYS(30294)
0517  7656                  ;**************************************
0518  7656                  ;
0519  7656  48             LVERT  PHA              ;SAVE REGISTERS
0520  7657  98                    TYA
0521  7658  48                    PHA
0522  7659  8A                    TXA
0523  765A  48                    PHA
0524  765B  20 00 74              JSR ADR1
0525  765E  A0 00                 LDY #$00
0526  7660  A5 58           LVER1  LDA T5
0527  7662  C9 08                 CMP #$08          ;LENGTH > 8 ?
0528  7664  30 17                 BMI LVER2
0529  7666  E9 08                 SBC #$08          ;REDUCE LENGTH BY 8 AND
0530  7668  85 58                 STA T5
0531  766A  A9 20                 LDA #$20          ;DRAW RVS SPACE
0532  766C  91 54                 STA (T1),Y
0533  766E  18                    CLC
0534  766F  A5 54                 LDA T1            ;SUBTRACT 40 FROM ADDRESS
0535  7671  E9 27                 SBC #$27
0536  7673  85 54                 STA T1
0537  7675  A5 55                 LDA T2
0538  7677  E9 00                 SBC #$00
0539  7679  85 55                 STA T2
0540  767B  D0 E3                 BNE LVER1
0541  767D  AA             LVER2  TAX
0542  767E  BD 89 76              LDA LVER3,X
0543  7681  91 54                 STA (T1),Y
0544  7683  68                    PLA
0545  7684  AA                    TAX
0546  7685  68                    PLA
0547  7686  A8                    TAY
0548  7687  68                    PLA
0549  7688  60                    RTS
0550  7689                  ;
0551  7689                  ;
0552  7689  64             LVER3  .BYTE $64,$52,$46,$40
0552  768A  52
0552  768B  46
0552  768C  40
0553  768D  43                    .BYTE $43,$44,$45,$63,$20
0553  768E  44
0553  768F  45
0553  7690  63
0553  7691  20
0554  7692                  ;
```

Bytes = 60

Fine Resolution Line Plotting - Vertical, Top to Bottom.

This routine draws a vertical bar of space characters from top to bottom terminated by a fine resolution horizontal line one eighth of a character wide. The top coordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of one eighth's of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

86 .... column number of start of fine resolution line plot (top end).

87 .... line number of start of fine resolution plot.

88 .... length of fine resolution line plot in one eighth's of a character space.

The routine is called from a Basic program with the command:

```
SYS(30409)
```

The following is an example of a Basic program using this routine:

```
10 POKE 86,10 : REM START AT COLUMN #10
20 POKE 87,10 : REM ON LINE # 10
30 POKE 88,60 : REM LINE LENGTH 60/8 CHARACTERS
40 SYS(30409) : REM CALL SUBROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0599  76C9                ;**************************************
0600  76C9                ;* DRAW VERTICAL BAR - TOP TO BOTTOM.
0601  76C9                ;* OF SPACES TERMINATED IN A THIN LINE
0602  76C9                ;* RESOLUTION 135 X 40. COLUMN # IN $56
0603  76C9                ;* LINE # IN $57, LENGTH IN $58
0604  76C9                ;* ROUTINE CALLED BY - SYS(30409)
0605  76C9                ;**************************************
0606  76C9                ;
0607  76C9  48            LTVERT PHA                  ;SAVE REGISTERS
0608  76CA  98                   TYA
0609  76CB  48                   PHA
0610  76CC  8A                   TXA
0611  76CD  48                   PHA
0612  76CE  20 00 74             JSR ADR1
0613  76D1  A0 00                LDY #$00
0614  76D3  A5 58       LTVER1  LDA T5
0615  76D5  C9 08                CMP #$08            ;LENGTH > 8 ?
0616  76D7  30 17                BMI LTVER2
0617  76D9  E9 08                SBC #$08            ;REDUCE LENGTH BY 8 AND
0618  76DB  85 58                STA T5
0619  76DD  A9 20                LDA #$20            ;DRAW RVS SPACE
0620  76DF  91 54                STA (T1),Y
0621  76E1  18                  CLC
0622  76E2  A5 54                LDA T1              ;ADD 40 TO ADDRESS
0623  76E4  69 28                ADC #$28
0624  76E6  85 54                STA T1
0625  76E8  A5 55                LDA T2
0626  76EA  69 00                ADC #$00
0627  76EC  85 55                STA T2
0628  76EE  D0 E3                BNE LTVER1
0629  76F0  AA          LTVER2  TAX
0630  76F1  BD FC 76             LDA LTVER3,X
0631  76F4  91 54                STA (T1),Y
0632  76F6  68                   PLA
0633  76F7  AA                   TAX
0634  76F8  68                   PLA
0635  76F9  A8                   TAY
0636  76FA  68                   PLA
0637  76FB  60                   RTS
0638  76FC              ;
0639  76FC              ;
0640  76FC  63          LTVER3  .BYTE $63,$45,$44,$43,$40
0640  76FD  45
0640  76FE  44
0640  76FF  43
0640  7700  40
0641  7701  46                  .BYTE $46,$52,$64,$20,$20
0641  7702  52
0641  7703  64
0641  7704  20
0641  7705  20
0642  7706
```

Bytes = 61.

EXAMPLE 3

```
10 REM ************************************
20 REM *THIS PROGRAM EMULATES THE DISPLAY OF
30 REM *AN ANAOLG EDGE METER, USE KEYS 4 & 6
40 REM *TO MOVE METER NEEDLE UP OR DOWN.
50 REM *ROUTINE USES THE FOLLOWING MACHINE CODE
60 REM *ROUTINES:
70 REM * SYS(31243) - CURSOR CONTROL
80 REM * SYS(30239) - FINE RES HORIZONTAL
90 REM * SYS(30090) - BORDER
100 REM * SYS(30029) - HORIZONTAL LINE
110 REM ************************************
120 POKE48,0:POKE49,112
130 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
140 PRINT"□"
145 REM
150 REM *DRAW SCALE IN INCREMENTS OF 10 FINE RES LINES
155 REM
160 FORQ=130TO0 STEP -10
170 POKE 86,10:POKE87,10:POKE88,Q:SYS(30239)
180 NEXTQ
183 REM
185 REM *DISPLAY TEXT USING CURSOR CONTROL
187 REM
190 DATA 6,11,"■ANALOG EDGE METER■"
200 DATA 8,9,0
210 DATA 8,28,130
220 DATA 20,7,"■DIGITAL VALUE ON METER■"
230 FORQ=1TO4
240 READA:POKE84,A:READA:POKE85,A:SYS(31243)
250 READA$:PRINTA$
260 NEXTQ
270 REM
280 REM *DRAW BORDERS AROUND METER DISPLAY
290 REM
320 POKE86,8:POKE87,9:POKE88,21:POKE89,4:SYS(30090)
340 POKE86,6:POKE87,5:POKE88,25:POKE89,9:SYS(30090)
350 REM
355 REM *SEE WHICH KEY PRESSED (LOC 151) IF KEY = 4 (151=41)
356 REM *THEN DISPLAY VALUE DECREASED, IF KEY = 6 (151=42)
357 REM *THEN DISPLAY VALUE INCREASED.
360 REM
370 A=0
380 X=PEEK(151)
390 IFX=41THEN A=A+1:GOTO420
400 IFX=42THEN A=A-1:GOTO420
410 GOTO380
420 IFA<0THENA=0
430 IFA>130THENA=130
434 REM
435 REM *LOCATE CURSOR TO DISPLAY DIGITAL EQUIVALENT
436 REM
440 POKE84,20:POKE85,30:SYS(31243)
450 PRINT"    ■■■■■";A
```

```
 454 REM
 455 REM *ERASE PREVIOUS METER NEEDLE DISPLAY
 456 REM
 460 POKE86,10:POKE87,11:POKE88,18:POKE00,32:SYS(30029)
 464 REM
 465 REM *DISPLAY NEW NEEDLE POSITION
 466 REM
 470 POKE86,10:POKE87,11:POKE88,A:SYS(30239)
 480 GOTO380
READY.
```

EXAMPLE 4

```
10 REM ************************************
20 REM *USING FINE RESOLUTION LINES THIS PROGRAM
30 REM *PLOTS THE GRAPH OF A FUNCTION.
40 REM *PROGRAM USES THE FOLLOWING MACHINE CODE ROUTINES:
50 REM *  SYS(30090) - DRAW BORDER
60 REM *  SYS(30294) - VERTICAL FINE RES LINE
70 REM *  SYS(31243) - CURSOR CONTROL
80 REM ************************************
90 POKE48,0:POKE49,112
100 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
110 PRINT"□"
120 REM
130 REM *DRAW VERTICAL SCALE IN INCREMENTS OF 10 FINE RES LINES
140 REM
150 FORQ=110TO0STEP-10
160 POKE86,0:POKE87,20:POKE88,Q:SYS(30294)
170 NEXTQ
180 REM
190 REM *DRAW BORDER AROUND DISPLAY
200 REM
210 POKE86,0:POKE87,6:POKE88,39:POKE89,16:SYS(30090)
220 REM
230 REM *DISPLAY TEXT USING CURSOR CONTROL
240 REM
250 POKE84,6:POKE85,10:SYS(31243)
260 PRINT"■GRAPH OF SINE FUNCTION■"
270 REM
280 REM *PLOT GRAPH OF FUNCTION. NOTE OFFSET GIVEN TO FUNCTION
290 REM *TO ENSURE NO NEGATIVE VALUES FOR PLOT OR VALUES OVER 135
300 REM
310 FORQ=2TO38
320 A=INT(SIN((Q-2)/5.5)*40)+50
330 POKE86,Q:POKE87,20:POKE88,A:SYS(30294)
340 NEXTQ
350 GETA$:IFA$=""THEN350:REM PRESS ANY KEY TO END
360 END
READY.
```

EXAMPLE 5

```
10 REM ************************************
20 REM *A SINGLE DATA SET OF UP TO 31 VALUES ALL +VE NUMBERS
30 REM *ARE DISPLAYED BY THIS ROUTINE AS A BARCHART.
40 REM *MACHINE CODE ROUTINES USED ARE:
50 REM *  SYS(30090) - DRAW BORDER
60 REM *  SYS(31243) - CURSOR CONTROL
70 REM *  SYS(29854) - PLOT VERTICAL FINE RES BAR
80 REM ************************************
90 POKE 48,0:POKE49,112
100 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
110 DIMP%(31)
120 REM
130 REM *DATA TO BE DISPLAYED
140 REM
150 DATA 2,5,6,3,9,8,10,15,11,17,24,22,18,16,17,18,19,20,21,18
160 DATA 25,23,28,26,20,13,5,21,28,26,35
170 REM
180 REM *PUT DATA INTO ARRAY P%
190 REM
200 FORQ=1TO31
210 READA
220 P%(Q)=A
230 NEXTQ
240 PRINT"■"
250 REM
260 REM *DISPLAY TEXT USING CURSOR CONTROL
270 REM
280 DATA2,14,"■EXAMPLE BAR CHART■"
290 DATA21,2,0
300 DATA23,7,"0.............................31"
310 FORQ=1TO3
320 READA:POKE84,A:READA:POKE85,A:SYS(31243)
330 READA$:PRINTA$
340 NEXTQ
350 REM
360 REM *DRAW BORDER AROUND DISPLAY AREA
370 REM
380 POKE86,5:POKE87,3:POKE88,33:POKE89,20:SYS(30090)
390 REM
400 REM
410 REM *DEFINE SCALE FOR PLOTTING
420 B=0
430 FORQ=1TO31
440 A=P%(Q)
450 IFA>BTHENB=A
460 NEXTQ
470 REM
480 REM *PRINT MAXIMUM DISPLAYED VALUE
490 REM
500 PRINT"■0000000"B
510 REM
520 REM *DISPLAY BARCHART USING VERTICAL BARPLOT ROUTINE
530 REM
540 A=B/135
550 FORQ=1TO31
```

```
 560 AS=INT(P%(Q)/A)
 570 POKE87,21:POKE86,Q+6:POKE88,AS
 580 SYS(29854)
 590 NEXTQ
 600 GETA$:IFA$=""THEN 600:REM **WAIT TILL KEY PRESSED TO END**
READY.
```

EXAMPLE 6

```
10 REM ************************************
20 REM *TWO DATA SETS EACH OF UP TO 15 VALUES ALL +VE NUMBERS
30 REM *ARE DISPLAYED BY THIS ROUTINE AS A BARCHART.
40 REM *MACHINE CODE ROUTINES USED ARE:
50 REM *  SYS(30090) - DRAW BORDER
60 REM *  SYS(31243) - CURSOR CONTROL
70 REM *  SYS(29854) - PLOT VERTICAL FINE RES BAR
75 REM *  SYS(30294) - PLOT VERTICAL FINE RES LINE
80 REM ************************************
90 POKE 48,0:POKE49,112
100 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
110 DIMP%(31)
120 REM
130 REM *DATA TO BE DISPLAYED
140 REM
150 DATA 2,1,4,2,6,3,8,4,10,5,12,6,14,7,16,8,18,9,20,10
160 DATA 22,11,24,12,26,13,28,14,30,15
170 FORQ=1TO30
180 READA
190 P%(Q)=A
200 NEXTQ
210 REM
220 REM *DISPLAY TEXT USING CURSOR CONTROL
230 REM
240 PRINT"□"
250 DATA1,14,"■EXAMPLE BAR CHART■"
260 DATA20,2,0
270 DATA22,7,"1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 "
280 DATA23,7,"                  0 1 2 3 4 5"
290 FORQ=1TO4
300 READA:POKE84,A:READA:POKE85,A:SYS(31243)
310 READA$:PRINTA$
320 NEXTQ
330 REM
340 REM *DRAW BORDER AROUND DISPLAY
350 REM
360 POKE86,5:POKE87,2:POKE88,32:POKE89,20:SYS(30090)
370 REM
380 REM *DEFINE SCALE FOR PLOTTING
390 REM
400 B=0
410 FORQ=1TO30
420 A=P%(Q)
430 IFA>BTHENB=A
440 NEXTQ
450 REM
460 REM *PRINT MAXIMUM DISPLAYED VALUE
470 REM
```

```
480 PRINT"■■■■■■■"B
490 A=B/135
500 FORQ=1TO30
510 REM
520 REM *DISPLAY BARCHART #1 (BAR)
530 REM
540 AS=INT(P%(Q)/A)
550 POKE87,20:POKE86,Q+6:POKE88,AS
560 SYS(29854)
570 REM
580 REM *DISPLAY BARCHART #2 (LINE)
590 REM
600 Q=Q+1
610 AS=INT(P%(Q)/A)
620 POKE87,20:POKE86,Q+6:POKE88,AS
630 SYS(30294)
640 NEXTQ
650 GETA$:IFA$=""THEN 650:REM **WAIT TILL KEY PRESSED TO END**
READY.
```

EXAMPLE 7

```
10 REM ************************************
20 REM *A SINGLE DATA SET OF UP TO 30 +VE OR -VE NUMBERS
30 REM *ARE DISPLAYED BY THIS ROUTINE AS A BARCHART.
40 REM *MACHINE CODE ROUTINES USED ARE:
50 REM *  SYS(30090) - DRAW BORDER
60 REM *  SYS(31243) - CURSOR CONTROL
70 REM *  SYS(29854) - PLOT VERTICAL FINE RES BAR UP
80 REM *  SYS(29969) - PLOT VERTICAL FINE RES BAR DOWN
90 REM ************************************
100 POKE 48,0:POKE49,112
110 POKE 52,0:POKE53,112:CLR:REM LIMIT TOP OF MEMORY
120 DIMP%(31)
130 REM
140 REM *DATA TO BE DISPLAYED
150 REM
160 DATA 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,-1,-2,-3,-4
170 DATA -5,-6,-7,-8,-9,-10,-11,-12,-13,-14
180 FORQ=1TO30
190 READA
200 P%(Q)=A
210 NEXTQ
220 REM
230 REM *DISPLAY TEXT USING CURSOR CONTROL
240 REM
250 PRINT"■"
260 DATA1,14,"■EXAMPLE BAR CHART■"
270 DATA12,2,0
280 DATA23,7,"1.....30"
290 FORQ=1TO3
300 READA:POKE84,A:READA:POKE85,A:SYS(31243)
310 READA$:PRINTA$
320 NEXTQ
330 REM
340 REM *DRAW BORDER AROUND DISPLAY
350 REM
```

```
360 POKE86,5:POKE87,2:POKE88,32:POKE89,21:SYS(30090)
370 REM
380 REM *DETERMINE AND DISPLAY MAX AND MIN DATA VALUES
390 REM
400 B=0
410 FORQ=1TO30
420 A=ABS(P%(Q))
430 IFA>BTHENB=A
440 NEXTQ
450 PRINT"■■■■■■"B
460 PRINT"■■■■■■
```

Notes

4 SCREEN & CHARACTER BLOCK SCROLLING

Section 4

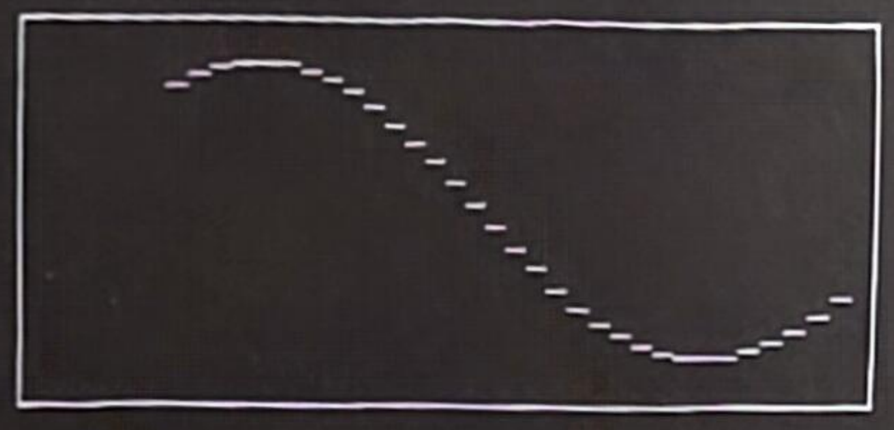
Screen and character block scrolling - 93
Scroll screen block right one column - 94
Scroll screen block left one column - 96
Scroll screen block up one line - 98
Scroll screen block down one line - 100
Example showing scrolling of long list of data - 103
Example plots graph of function and scrolles it left - 104

EXAMPLE OF SCROLLED [illegible]

USE KEY 8 TO SCROLL UP
USE KEY 2 TO SCROLL DOWN

ITEM 7
ITEM 8
ITEM 9
ITEM 10
ITEM 11
ITEM 12

EXAMPLE DISPLAY OF LONG WAVEFORM
USING LEFT SCROLLING



ABS
CLR

CLR
ABS

ABS
CLR
DATA
DIM
END
FOR
GOSUB
GOTO
IF

IF
GOTO
GOSUB
FOR
END
DIM
DATA
CLR
ABS

SCREEN & CHARACTER BLOCK SCROLLING 4

Scrolling is a feature with which all PET users are familiar, the upward scrolling of the entire screen being a standard feature of the PET. The entire screen contents is moved up one line, leaving a blank line at the bottom of the screen and erasing the top line. Usefull when listing programs but otherwise is not of great value. For a scrolling feature to be of value the scrolling should be both up, down, left and right, and should not just be for the whole screen but allow small blocks of the data to be scrolled inependently of the rest of the screen. This facility would allow the display of long lists of data or graphs which are too long to fit the PET screen. Such displays could be scrolled left and right or up and down by the user and compared against the non scrolled fixed portion of the screen where coordinates or other reference data are located. Examples of some of the many uses of screen scrolling are shown in the example programs at the end of this section.

Scrolling is easily performed on a memory mapped video display since it entails just the movement of an entire block of memory by a few bytes. To scroll up by one line each memory location in the scrolled block is moved down in memory by 40 bytes (80 bytes in an 80 column machine). Similarly to scroll right one column all memory locations in the block are moved up one byte. The top and bottom lines or right and left columns must of course be treated differently since if the screen block is being scrolled up the top line of the block must disappear and the bottom line is filled with space characters. It is not practical to write a program in Basic to perform a scrolling function, it is far too slow, machine code routines are therefore essential.

Scroll Screen Block Right One Column.

This routine scrolls a specified block of screen characters one column to the right, the left most column is filled with spaces and the right most column disappears. The top left coordinates of the scrolled block are determined by two variables, column number and line number. The width and height of the block are set by variables with a minimum of 0 and a maximum equivalent to the screen dimensions. The variable locations required by the routine are as follows:

86 .... column number of top left of scrolled block.
87 .... line number of top left of scrolled block.
88 .... height of scrolled block.
89 .... width of scrolled block.

The routine is called from a Basic program with the command:

SYS(30723)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  :REM TOP LEFT OF BLOCK AT COLUMN #5
 20 POKE 87,10 :REM AND LINE #10
 30 POKE 88,10 :REM SCROLLED BLOCK 10 LINES HIGH
 40 POKE 89,30 :REM AND 30 CHARACTERS WIDE
 50 SYS(30723) :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0848  7803                ;**************************************
0849  7803                ;* SCROLL CONTENTS OF SCREEN BLOCK
0850  7803                ;* RIGHT ONE COLUMN. TOP LEFT COORDINATES
0851  7803                ;* OF BLOCK IN $56 (COL) AND $57 (LINE)
0852  7803                ;* BLOCK HEIGHT IN $58 AND WIDTH IN $59
0853  7803                ;* ROUTINE CALLED BY - SYS(30723)
0854  7803                ;**************************************
0855  7803                ;
0856  7803                ;
0857  7803  48            RSCROL PHA
0858  7804  98                   TYA
0859  7805  48                   PHA
0860  7806  8A                   TXA
0861  7807  48                   PHA
0862  7808  20 00 74             JSR ADR1
0863  780B  38                   SEC
0864  780C  A5 54                LDA T1
0865  780E  E9 01                SBC #$01          ;CALCULATE OFFSET ADDRESS
0866  7810  85 56                STA T3
0867  7812  A5 55                LDA T2
0868  7814  E9 00                SBC #$00
0869  7816  85 57                STA T4
0870  7818  A6 58                LDX T5
0871  781A  A4 59       RSCRL1   LDY T6
0872  781C  88                   DEY
0873  781D  B1 56       RSCRL2   LDA (T3),Y        ;SCROLL RIGHT
0874  781F  91 54                STA (T1),Y
0875  7821  88                   DEY
0876  7822  D0 F9                BNE RSCRL2
0877  7824  A9 20                LDA #$20          ;FILL RIGHT HAND COLUMN
0878  7826  91 54                STA (T1),Y        ;WITH BLANKS
0879  7828  18                   CLC
0880  7829  A5 54                LDA T1
0881  782B  69 28                ADC #$28          ;ADDRESS FOR NEXT ROW
0882  782D  85 54                STA T1
0883  782F  90 02                BCC RSCRL3
0884  7831  E6 55                INC T2
0885  7833  18          RSCRL3   CLC
0886  7834  A5 56                LDA T3
0887  7836  69 28                ADC #$28
0888  7838  85 56                STA T3
0889  783A  90 02                BCC RSCRL4
0890  783C  E6 57                INC T4
0891  783E  CA          RSCRL4   DEX
0892  783F  D0 D9                BNE RSCRL1
0893  7841  68                   PLA
0894  7842  AA                   TAX
0895  7843  68                   PLA
0896  7844  A8                   TAY
0897  7845  68                   PLA
0898  7846  60                   RTS
0899  7847
```

Byte=68

Scroll Screen Block Left One Column.

This routine scrolls a specified block of screen characters one column to the left, the right most column is filled with spaces and the left most column disappears. The top left coordinates of the scrolled block are determined by two variables, column number and line number. The width and height of the block are set by variables with a minimum of 0 and a maximum equivalent to the screen dimensions. The variable locations required by the routine are as follows:

86 .... column number of top left of scrolled block.
87 .... line number of top left of scrolled block.
88 .... height of scrolled block.
89 .... width of scrolled block.

The routine is called from a Basic program with the command:

SYS(30791)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  :REM TOP LEFT OF BLOCK AT COLUMN #5
 20 POKE 87,10 :REM AND LINE #10
 30 POKE 88,10 :REM SCROLLED BLOCK 10 LINES HIGH
 40 POKE 89,30 :REM AND 30 CHARACTERS WIDE
 50 SYS(30791) :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0901  7847                ;**********************************
0902  7847                ;* SCROLL CONTENTS OF SCREEN BLOCK
0903  7847                ;* LEFT ONE COLUMN. TOP LEFT COORDINATES
0904  7847                ;* OF BLOCK IN $56 (COL) AND $57 (LINE)
0905  7847                ;* BLOCK HEIGHT IN $58 AND WIDTH IN $59
0906  7847                ;* ROUTINE CALLED BY - SYS(30791)
0907  7847                ;**********************************
0908  7847                ;
0909  7847                ;
0910  7847  48     LSCROL PHA
0911  7848  98            TYA
0912  7849  48            PHA
0913  784A  8A            TXA
0914  784B  48            PHA
0915  784C  20 00 74       JSR ADR1
0916  784F  18            CLC
0917  7850  A5 54          LDA T1              ;CALCULATE OFFSET
0918  7852  69 01          ADC #$01
0919  7854  85 56          STA T3
0920  7856  A5 55          LDA T2
0921  7858  69 00          ADC #$00
0922  785A  85 57          STA T4
0923  785C  A6 58          LDX T5
0924  785E  A0 00    LSCRL1 LDY #$00
0925  7860  B1 56    LSCRL2 LDA (T3),Y         ;SCROLL LEFT
0926  7862  91 54          STA (T1),Y
0927  7864  C8            INY
0928  7865  C4 59          CPY $59
0929  7867  90 F7          BCC LSCRL2
0930  7869  A9 20          LDA #$20            ;FILL RIGHT HAND COL
0931  786B  91 54          STA (T1),Y          ;WITH SPACES
0932  786D  18            CLC
0933  786E  A5 54          LDA T1              ;NEXT ROW ADDRESS
0934  7870  69 28          ADC #$28
0935  7872  85 54          STA T1
0936  7874  90 02          BCC LSCRL3
0937  7876  E6 55          INC T2
0938  7878  18      LSCRL3 CLC
0939  7879  A5 56          LDA T3
0940  787B  69 28          ADC #$28
0941  787D  85 56          STA T3
0942  787F  90 02          BCC LSCRL4
0943  7881  E6 57          INC T4
0944  7883  CA      LSCRL4 DEX
0945  7884  D0 D8          BNE LSCRL1          ;NEXT LINE
0946  7886  68            PLA
0947  7887  AA            TAX
0948  7888  68            PLA
0949  7889  A8            TAY
0950  788A  68            PLA
0951  788B  60            RTS
0952  788C
```

Bytes = 69

Scroll Screen Block Up One Line.

This routine scrolls a specified block of screen characters up one line, the bottom line is filled with spaces and the top line disappears. The top left coordinates of the scrolled block are determined by two variables, column number and line number. The width and height of the block are set by variables with a minimum of 0 and a maximum equivalent to the screen dimensions. The variable locations required by the routine are as follows:

86 .... column number of top left of scrolled block.
87 .... line number of top left of scrolled block.
88 .... width of scrolled block.
89 .... height of scrolled block.

The routine is called from a Basic program with the command:

SYS(30552)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  :REM TOP LEFT OF BLOCK AT COLUMN #5
 20 POKE 87,10 :REM AND LINE #10
 30 POKE 88,30 :REM SCROLLED BLOCK 30 CHARACTERS WIDE
 40 POKE 89,10 :REM AND 10 LINES HIGH
 50 SYS(30552) :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

0722  7758                ;**************************************
0723  7758                ;* SCROLL CONTENTS OF SCREEN BLOCK
0724  7758                ;* UP ONE LINE. TOP LEFT COORDINATES
0725  7758                ;* OF BLOCK IN $56 (COL) AND $57 (LINE)
0726  7758                ;* BLOCK HEIGHT IN $59 AND WIDTH IN $58
0727  7758                ;* ROUTINE CALLED BY - SYS(30552)
0728  7758                ;**************************************
0729  7758                ;
0730  7758                ;
0731  7758  48          USCROL PHA
0732  7759  98                 TYA
0733  775A  48                 PHA
0734  775B  8A                 TXA
0735  775C  48                 PHA
0736  775D  20 00 74           JSR ADR1
0737  7760  18                 CLC
0738  7761  A5 54             LDA T1            ;CALCULATE START OF LINE
0739  7763  69 28             ADC #$28          ;BELOW TOP AND STORE
0740  7765  85 56             STA T3            ;ADDRESS IN $56 & $57
0741  7767  A5 55             LDA T2
0742  7769  69 00             ADC #$00
0743  776B  85 57             STA T4
0744  776D  A6 59             LDX T6
0745  776F  A4 58       USCRL1 LDY T5
0746  7771  88                 DEY
0747  7772  B1 56       USCRL2 LDA (T3),Y       ;MOVE CHARACTERS UP
0748  7774  91 54             STA (T1),Y        ;ONE LINE
0749  7776  88                 DEY
0750  7777  10 F9              BPL USCRL2
0751  7779  CA                 DEX
0752  777A  F0 19              BEQ USCRL5
0753  777C  18                 CLC
0754  777D  A5 54             LDA T1
0755  777F  69 28             ADC #$28          ;CALCULATE START ADDRESSES
0756  7781  85 54             STA T1            ;FOR NEXT PAIR OF LINES
0757  7783  90 02             BCC USCRL3        ;DOWN
0758  7785  E6 55             INC T2
0759  7787  18             USCRL3 CLC
0760  7788  A5 56                 LDA T3
0761  778A  69 28                 ADC #$28
0762  778C  85 56                 STA T3
0763  778E  90 02                 BCC USCRL4
0764  7790  E6 57                 INC T4
0765  7792  4C 6F 77       USCRL4 JMP USCRL1       ;NEXT LINE
0766  7795  A4 58       USCRL5 LDY T5
0767  7797  88                 DEY
0768  7798  A9 20       USCRL6 LDA #$20         ;CLEAR BOTTOM LINE
0769  779A  91 54             STA (T1),Y
0770  779C  88                 DEY
0771  779D  10 F9              BPL USCRL6
0772  779F  68                 PLA
0773  77A0  AA                 TAX
0774  77A1  68                 PLA
0775  77A2  A8                 TAY
0776  77A3  68                 PLA
0777  77A4  60                 RTS
0778  77A5
```

Bytes = 77.

Scroll Screen Block Down One Line.

This routine scrolls a specified block of screen characters down one line, the top line is filled with spaces and the bottom line disappears. The top left coordinates of the scrolled block are determined by two variables, column number and line number. The width and height of the block are set by variables with a minimum of 0 and a maximum equivalent to the screen dimensions. The variable locations required by the routine are as follows:

86 .... column number of top left of scrolled block.
87 .... line number of top left of scrolled block.
88 .... width of scrolled block.
89 .... height of scrolled block.

The routine is called from a Basic program with the command:

SYS(30629)

The following is an example of a Basic program using this routine:

```
 10 POKE 86,5  :REM TOP LEFT OF BLOCK AT COLUMN #5
 20 POKE 87,10 :REM AND LINE #10
 30 POKE 88,30 :REM SCROLLED BLOCK 30 CHARACTERS WIDE
 40 POKE 89,10 :REM AND 10 LINES HIGH
 50 SYS(30629) :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE              LINE

0780  77A5                    ;**************************************
0781  77A5                    ;* SCROLL CONTENTS OF SCREEN BLOCK
0782  77A5                    ;* DOWN ONE LINE. TOP LEFT COORDINATES
0783  77A5                    ;* OF BLOCK IN $56 (COL) AND $57 (LINE)
0784  77A5                    ;* BLOCK HEIGHT IN $59 AND WIDTH IN $58
0785  77A5                    ;* ROUTINE CALLED BY - SYS(30629)
0786  77A5                    ;**************************************
0787  77A5                    ;
0788  77A5                    ;
0789  77A5  48          DSCROL PHA
0790  77A6  98                 TYA
0791  77A7  48                 PHA
0792  77A8  8A                 TXA
0793  77A9  48                 PHA
0794  77AA  20 00 74           JSR ADR1
0795  77AD  A6 59             LDX T6
0796  77AF  CA                 DEX
0797  77B0  18          DSCRL1 CLC
0798  77B1  A5 54              LDA T1
0799  77B3  69 28              ADC #$28          ;CALCULATE BOTTOM LEFT
0800  77B5  85 54              STA T1            ;ADDRESS
0801  77B7  90 02              BCC DSCRL2
0802  77B9  E6 55              INC T2
0803  77BB  CA          DSCRL2 DEX
0804  77BC  D0 F2              BNE DSCRL1
0805  77BE  38                 SEC
0806  77BF  A5 54              LDA T1
0807  77C1  E9 28              SBC #$28          ;CALCULATE START OF NEXT
0808  77C3  85 56              STA T3            ;LINE UP AND STORE
0809  77C5  A5 55              LDA T2            ;IN $56 AND $57
0810  77C7  E9 00              SBC #$00
0811  77C9  85 57              STA T4
0812  77CB  A6 59              LDX T6
0813  77CD  A4 58    DSCRL3    LDY T5
0814  77CF  88                 DEY
0815  77D0  B1 56    DSCRL4    LDA (T3),Y         ;SCROLL DOWN
0816  77D2  91 54              STA (T1),Y        ;ONE LINE
0817  77D4  88                 DEY
0818  77D5  10 F9              BPL DSCRL4
0819  77D7  CA                 DEX
0820  77D8  F0 19              BEQ DSCRL7
0821  77DA  38                 SEC
0822  77DB  A5 54              LDA T1            ;NEXT PAIR OF LINES UP
0823  77DD  E9 28              SBC #$28
0824  77DF  85 54              STA T1
0825  77E1  B0 02              BCS DSCRL5
0826  77E3  C6 55              DEC T2
0827  77E5  38          DSCRL5 SEC
0828  77E6  A5 56              LDA T3
0829  77E8  E9 28              SBC #$28
0830  77EA  85 56              STA T3
0831  77EC  B0 02              BCS DSCRL6
0832  77EE  C6 57              DEC T4
0833  77F0  4C CD 77    DSCRL6 JMP DSCRL3
0834  77F3  A4 58    DSCRL7    LDY T5
```

Bytes: 96.

```
LINE# LOC    CODE          LINE

0835  77F5  88                    DEY
0836  77F6  A9 20       DSCRL8  LDA #$20          ;CLEAR TOP LINE
0837  77F8  91 54               STA (T1),Y
0838  77FA  88                    DEY
0839  77FB  D0 F9               BNE DSCRL8
0840  77FD  68                    PLA
0841  77FE  AA                    TAX
0842  77FF  68                    PLA
0843  7800  A8                    TAY
0844  7801  68                    PLA
0845  7802  60                    RTS
0846  7803
```

EXAMPLE 8

```
10 REM ************************************
20 REM *THIS PROGRAM USES THE UP AND DOWN BLOCK SCROLL
30 REM *ROUTINES TO DISPLAY THE CONTENTS OF A LONG LIST
40 REM *DATA ITEMS. USE KEY 8 TO SCROLL UP THE LIST
50 REM *AND KEY 2 TO SCROLL DOWN THE LIST.
60 REM *THE MACHINE CODE ROUTINES USED BY THIS PROGRAM ARE:
70 REM *    SYS(30090) - DRAW A BORDER
80 REM *    SYS(30552) - SCROLL BLOCK UP ONE LINE
90 REM *    SYS(30629) - SCROLL BLOCK DOWN ONE LINE
100 REM *    SYS(31243) - CURSOR CONTROL
110 REM ************************************
120 POKE48,0:POKE49,112
130 POKE52,0:POKE53,112:CLR:REM LOWER TOP OF MEMORY
140 DIML$(40)
150 REM
160 REM *TABLE OF DATA TO BE DISPLAYED, NOTE THE USE OF
170 REM *NULL ENTRIES AT BEGINING AND END OF THE TABLE
180 REM *THIS GIVES A BLANK SCREEN TRAILER AT BOTH ENDS
190 REM *OF THE DISPLAY, ESSENTIAL FOR NEATNESS.
200 REM
210 DATA" "," "," "," "," "," "," "," "
220 DATA "ITEM 1 ....................","ITEM 2 ...................."
230 DATA "ITEM 3 ....................","ITEM 4 ...................."
240 DATA "ITEM 5 ....................","ITEM 6 ...................."
250 DATA "ITEM 7 ....................","ITEM 8 ...................."
260 DATA "ITEM 9 ....................","ITEM 10 ..................."
270 DATA "ITEM 11 ...................","ITEM 12 ..................."
280 DATA "ITEM 13 ...................","ITEM 14 ..................."
290 DATA "ITEM 15 ...................","ITEM 16 ..................."
300 DATA "ITEM 17 ...................","ITEM 18 ..................."
310 DATA "ITEM 19 ...................","ITEM 20 ..................."
320 DATA" "," "," "," "," "," "," "," "
330 FORQ=1TO36
340 READA$
350 L$(Q)=A$
360 NEXTQ
370 REM
380 REM *DISPLAY TEXT USING CURSOR CONTROL
390 REM
400 PRINT"□"
410 DATA3,6,"■EXAMPLE OF SCROLLED LONG LIST■"
420 DATA7,9,"■USE KEY 8 TO SCROLL UP  ■"
430 DATA8,9,"■USE KEY 2 TO SCROLL DOWN■"
440 FORQ=1TO3
450 READA:POKE84,A:READA:POKE85,A:SYS(31243)
460 READA$:PRINTA$
470 NEXTQ
480 REM
490 REM *DRAW BORDER AROUND SCROLLED PORTION OF SCREEN
500 REM
510 POKE86,0:POKE87,10:POKE88,39:POKE89,10:SYS(30090)
520 GOTO680
530 REM
```

```
540 REM *LOOK FOR KEY INPUT IN LOCATION 151
550 REM *IF 151=50 THEN KEY 8 PRESSED AND SCROLL UP
560 REM *IF 151=18 THEN KEY 2 PRESSED AND SCROLL DOWN
570 REM *ALSO CHECK POINTER TO TABLE IS WITHIN BOUNDS.
580 REM
590 IFQ<1THENQ=1
600 IFQ>29THENQ=29
610 X=PEEK(151)
620 IFX=50 THEN Q=Q+1:GOTO680
630 IFX=18 THEN Q=Q-1:GOTO750
640 GOTO590
650 REM
660 REM *SCROLL LIST UP ONE ENTRY
670 REM
680 POKE86,2:POKE87,12:POKE88,37:POKE89,6:SYS(30552)
690 POKE84,17:POKE85,2:SYS(31243)
700 PRINTL$(Q+5)
710 GOTO590
720 REM
730 REM *SCROLL LIST DOWN ONE ENTRY
740 REM
750 POKE86,2:POKE87,12:POKE88,37:POKE89,6:SYS(30629)
760 POKE84,12:POKE85,2:SYS(31243)
770 PRINTL$(Q)
780 GOTO590
READY.
```

EXAMPLE 9

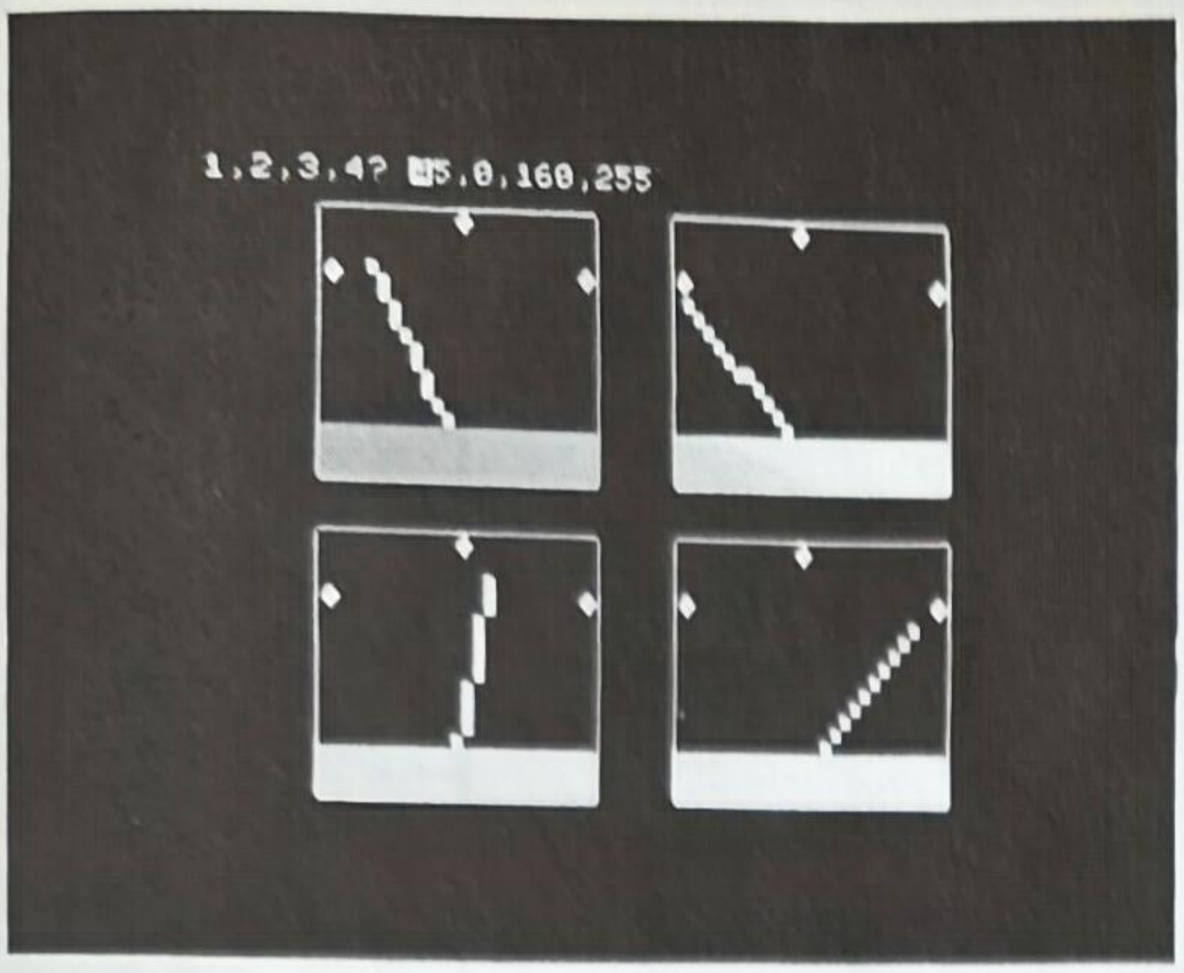
```
1 REM ************************************
2 REM *THIS IS A VARIATION ON EXAMPLE4. THE GRAPH OF THE
3 REM *FUNCTION IS SCROLLED ACCROSS THE SCREEN FROM RIGHT
4 REM *TO LEFT CONTINUOUSLY AS EACH PLOT POINT IS CALCULATED.
5 REM *IDEAL FOR DISPLAYING LONG GRAPHICAL DISPLAYS.
6 REM *USES THE FOLLOWING MACHINE CODE ROUTINES:
7 REM *  SYS(31243) - CURSOR CONTROL
8 REM *  SYS(30090) - DRAW BORDER
9 REM *  SYS(30791) - SCROLL SCREEN BLOCK LEFT
10 REM *  SYS(30294) - DRAW VERTICAL FINE RESOLUTION LINE
11 REM ************************************
18 POKE48,0:POKE49,112
20 POKE52,0:POKE53,112:CLR:REM LOWER TOP OF MEMORY POINTERS
30 PRINT"□"
34 REM
35 REM *DRAW BORDER AROUND SCROLLED SCREEN AREA
36 REM
40 POKE86,0:POKE87,7:POKE88,39:POKE89,15:SYS(30090)
44 REM
45 REM *DISPLAY TEXT USING CURSOR CONTROL
46 REM
50 POKE84,3:POKE85,4:SYS(31243)
60 PRINT"■EXAMPLE DISPLAY OF LONG WAVEFORM■"
70 POKE84,4:POKE85,10:SYS(31243)
80 PRINT"■USING LEFT SCROLLING■"
84 REM
85 REM *DISPLAY GRAPH OF FUNCTION SCROLLED FROM RIGHT TO LEFT
86 REM
```

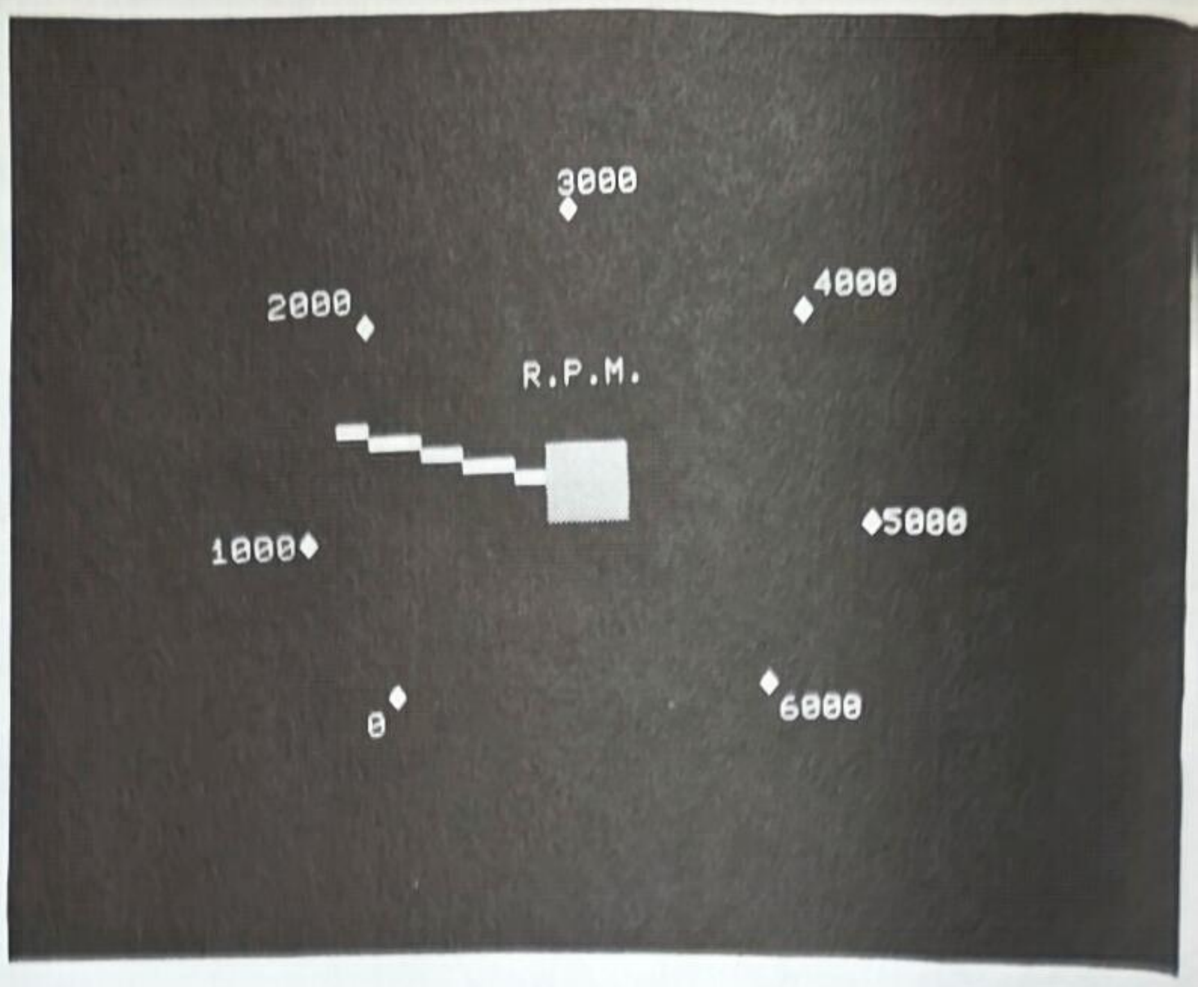
```
 90 FORQ=1TO100STEP.15
 100 A=INT(SIN(Q)*40)+45
 110 POKE86,2:POKE87,8:POKE88,12:POKE89,36:SYS(30791)
 120 POKE86,38:POKE87,19:POKE88,A:SYS(30294)
 130 FORX=1TO100:NEXTX
 140 NEXTQ
READY.
```

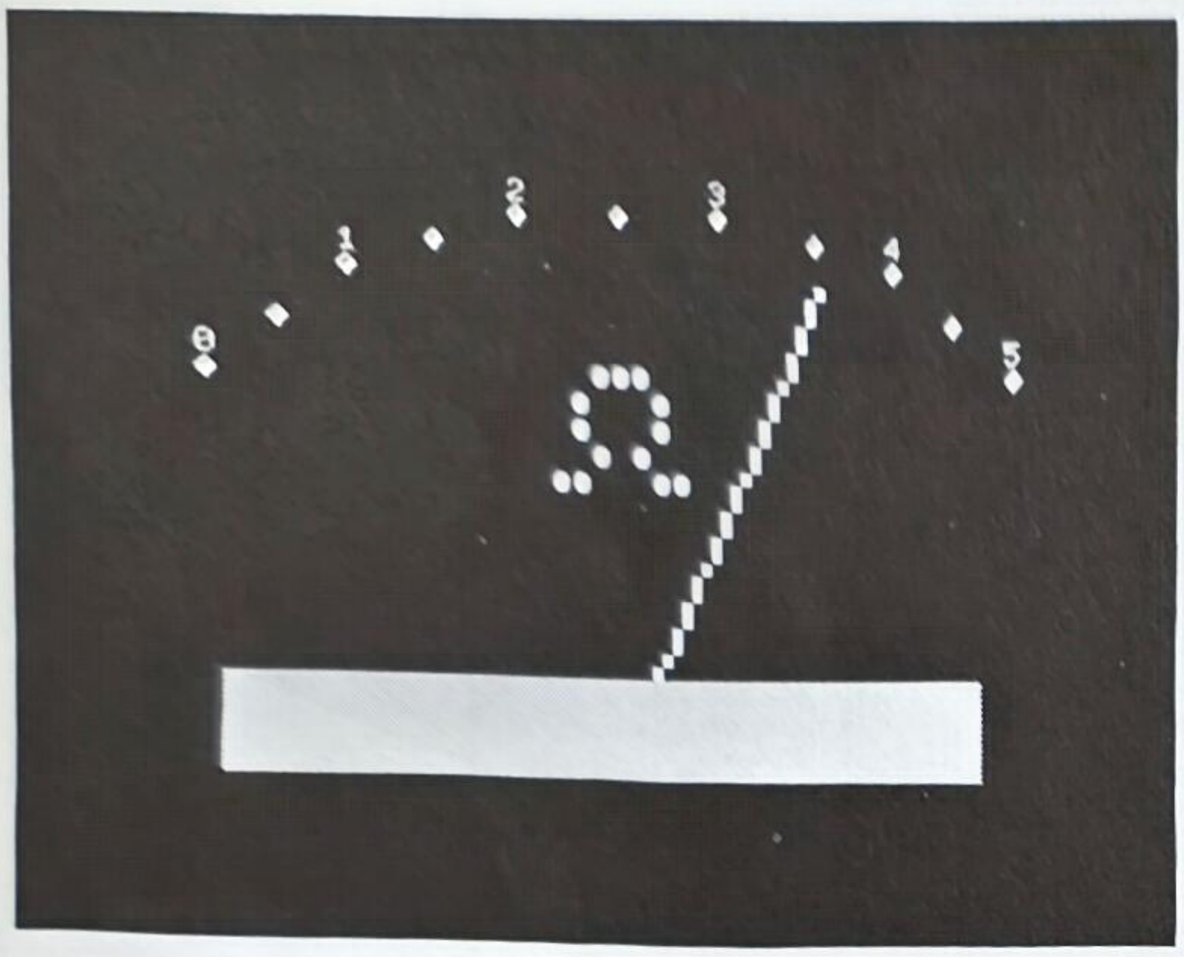
Notes

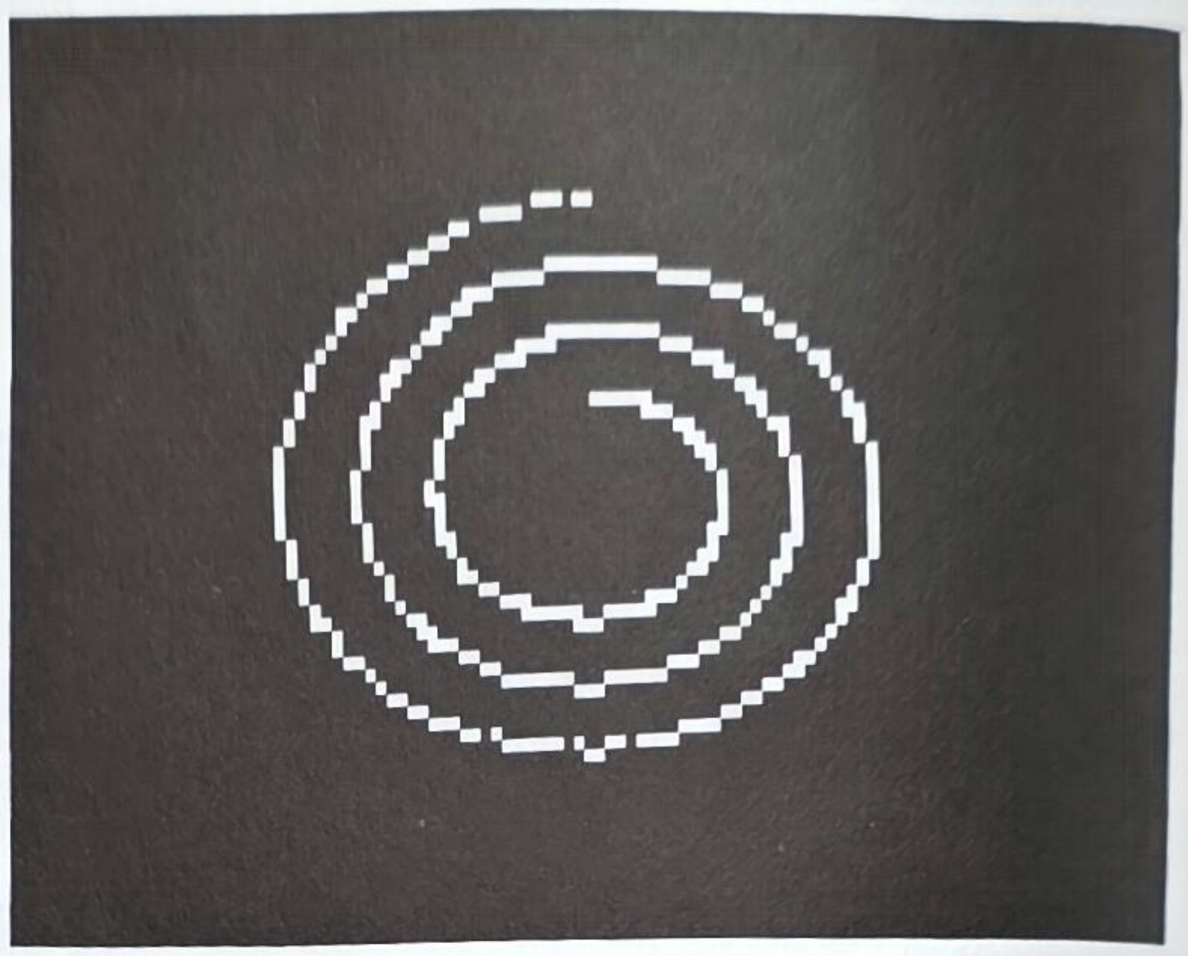
5 DOUBLE DENSITY PLOTTING

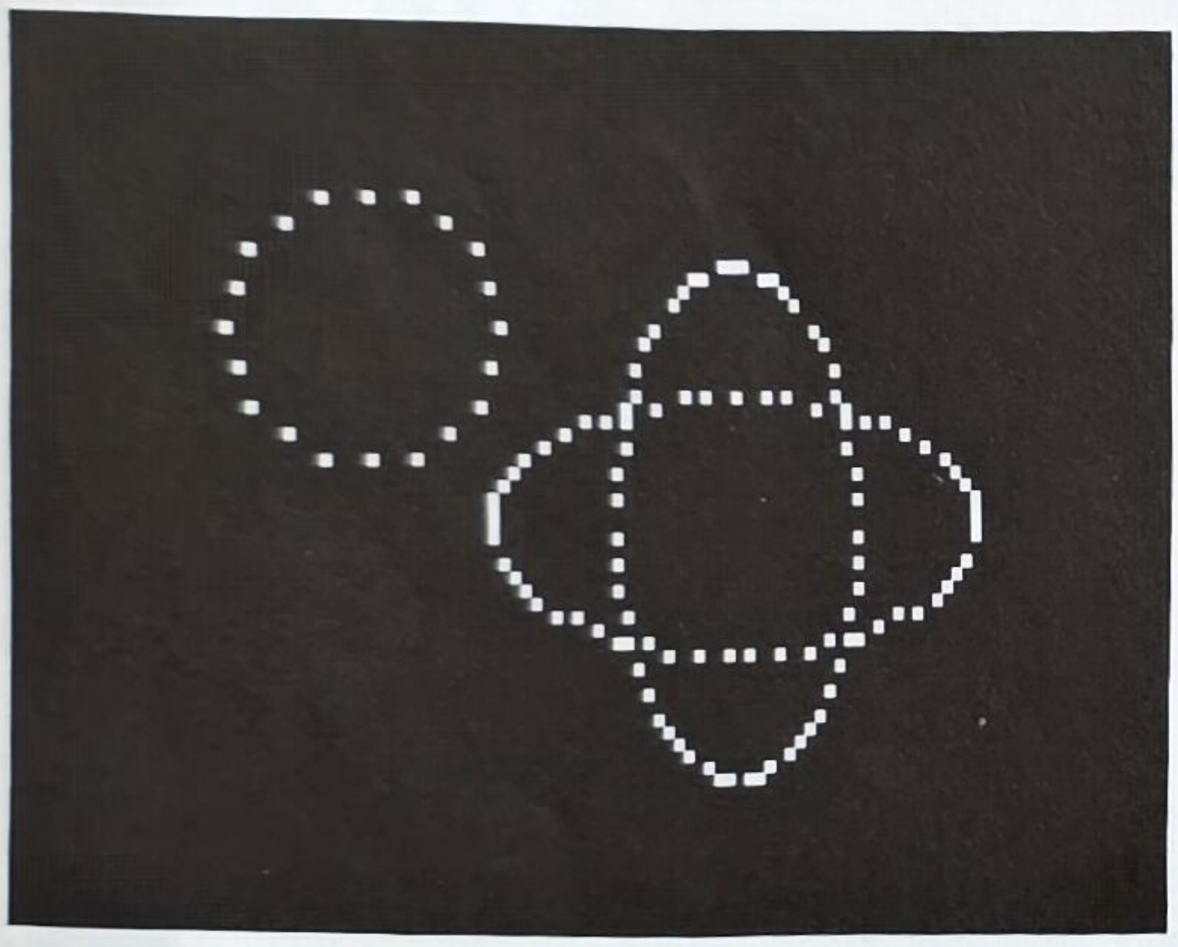
Double density point plotting - 109
Routine to plot points in double density - 2
Routine to plot lines in double density 117
Basic program to show principles of line plot - 122
Example plots a circle in double density points - 123
Double density used to plot a satellite orbit - 123
Double density used to interpolate a graph from data - 124
Double density used to display analog clock, keeps time - 125
Double density used to emulate analog panel meters - 129

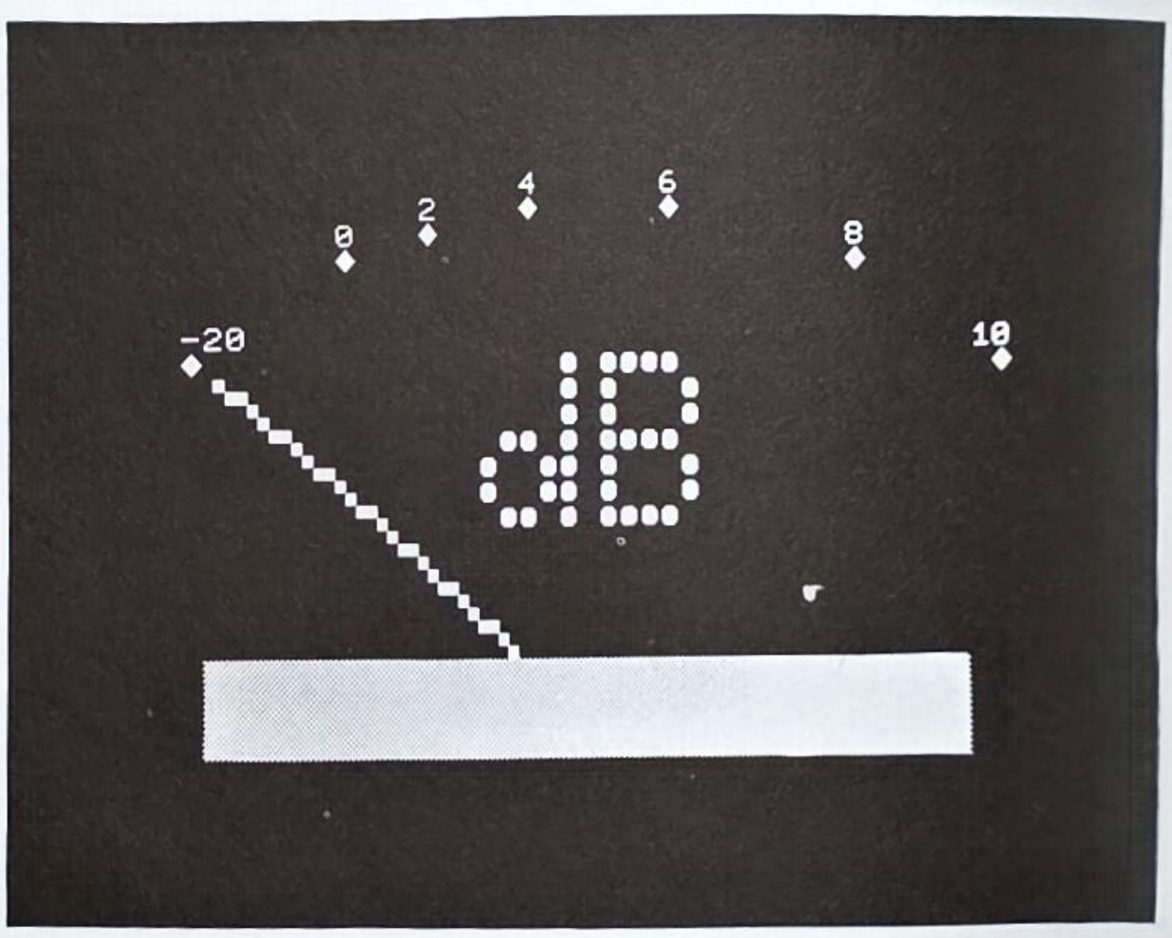

1,2,3,4? 5,0,160,255

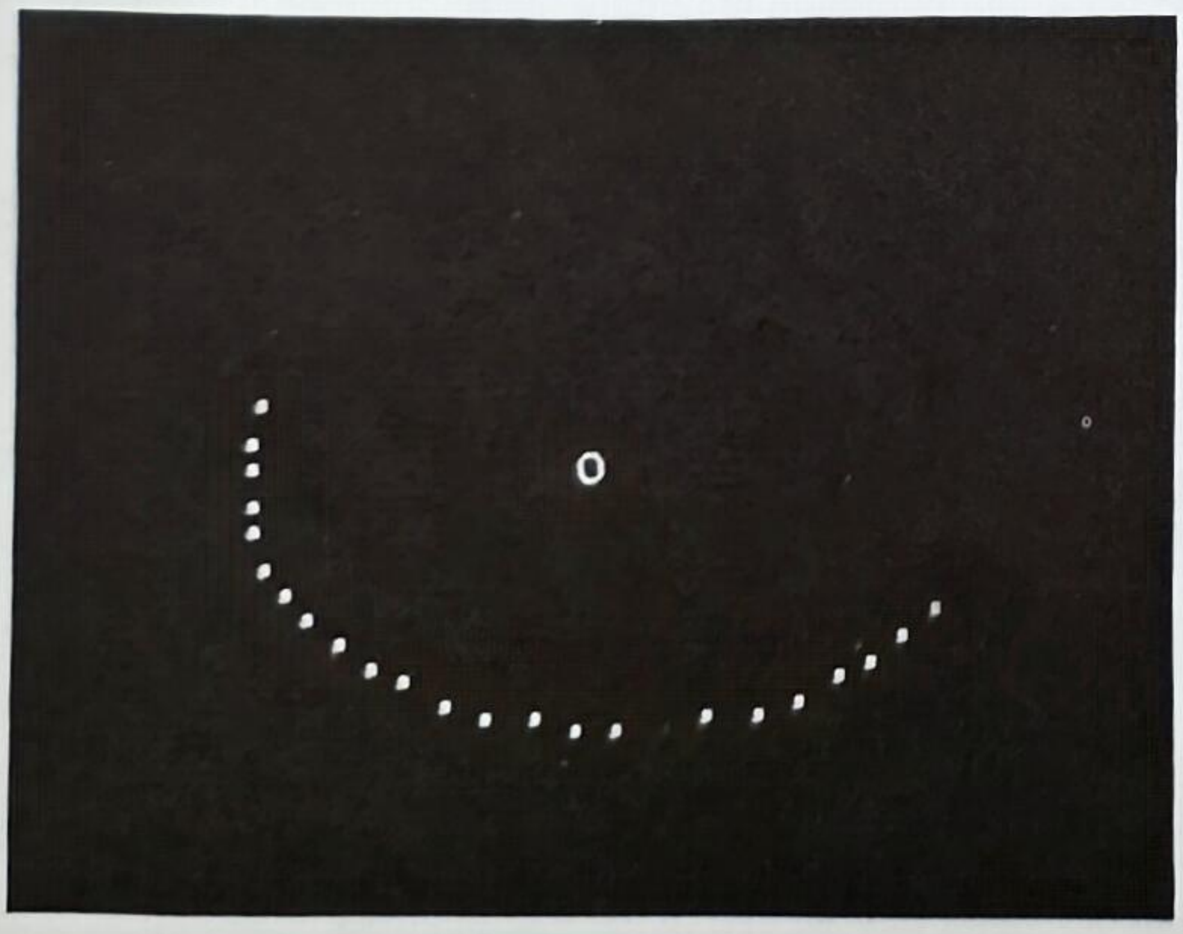

3000
2000
4000
R.P.M.
1000
5000
0
6000

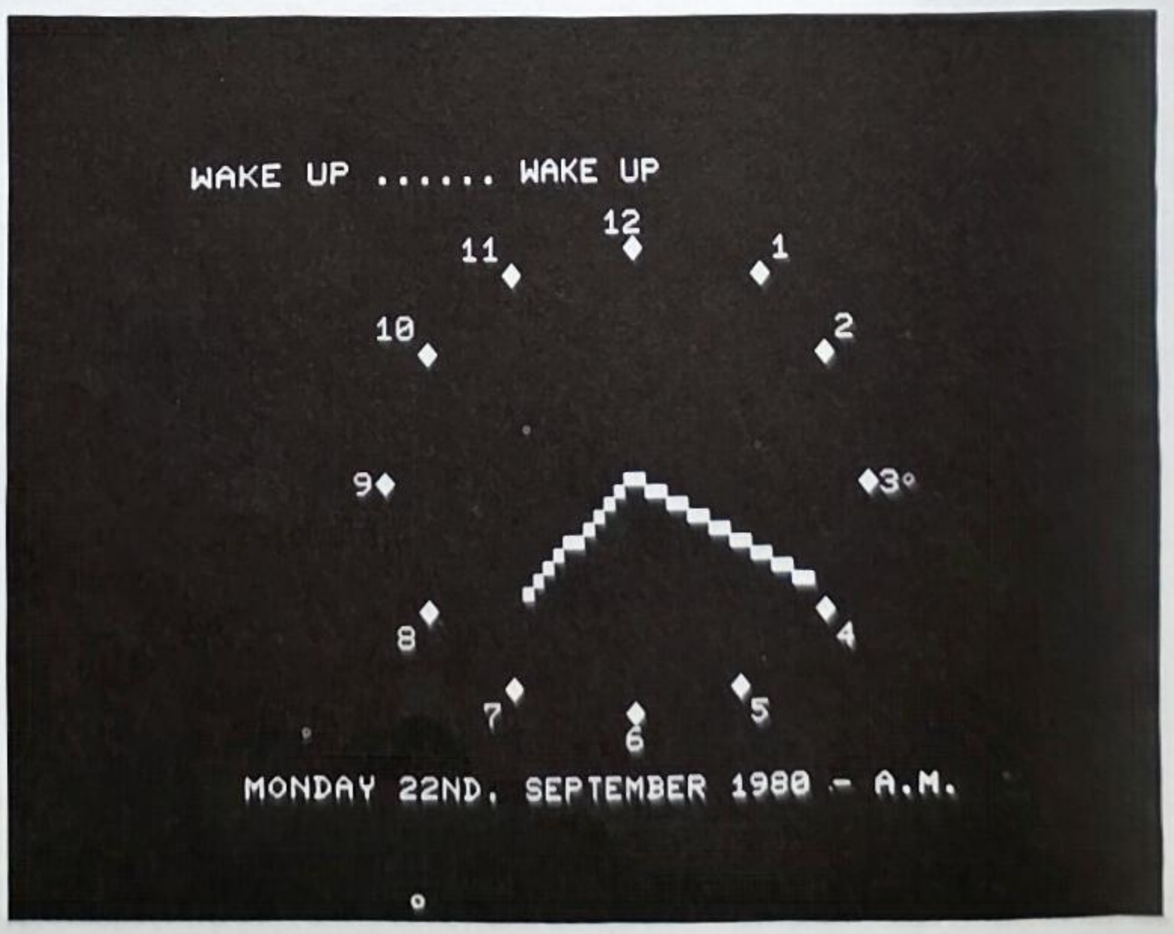

0
1
2
3
4
5






0
2
4
6
8
-20
10
dB


0


WAKE UP ..... WAKE UP
12
11
1
10
2
9
3
8
4
7
6
5
MONDAY 22ND, SEPTEMBER 1980 - A.M.

DOUBLE DENSITY PLOTTING 5

One drawback with having a display only 40 characters wide and 25 lines deep, is the poor definition acheivable when displaying data in graphical form. One dimensional high resolution plotting is the answer for barcharts and certain graphs, but not ideal in most other cases. Plotting in two dimensions is limited to single character point plotting and the number of characters per line cannot be increased without modifying the circuitry. However, by clever use of the fourteen different quarter square graphics characters the resolution of the screen can be improved from 40 x 25 to 80 x 50. The complete set of quarter square characters are shown in Fig 10 together with their hexadecimal and decimal ASCII code values. The principles behind a routine to plot in quarter squares is shown in the Basic program EXAMPLE 10. This uses just five of the sixteen characters and is therefore much simplified and cruder in its display. The double density point plot routine in machine code is very versatile, allowing any point on the 80 x 50 coordinates to be plotted as a quarter square character. The coordinate system is different to that used by most of the other routines in the book but conforms to the standard used for graph drawing, with coordinates 0,0 in the bottom left corner of the screen. The point plot routine is called by a second routine in the package, this draws a straight line on the screen between two sets of coordinates using double density characters. The machine code routine to do this follows the logic in the Basic program EXAMPLE 11 which performs the same function. The labels used in the source code listing of the double density line drawing program refer to the line numbers in the basic code which perform the same function.

The demonstration programs in this section cover a wide range of potential applications using either double density point plot or line drawing. The programs EXAMPLE 11, EXAMPLE 12, and EXAMPLE 13 show how point plotting can be used to draw lines, circles and elipses. The program EXAMPLE 13 shows a single quarter square point moving in an eliptical orbit around a point on the screen. Graph plotting accounts for the most frequent use of double density plotting. EXAMPLE 14 will take a set of values and draw a graph by interpolating between these values, the resulting graph can be any size and located anywhere on the screen. EXAMPLE 15 and EXAMPLE 16 show how double density plotting can be used to produce very effective displays in these two cases of analog devices, the hands of a clock (which keeps very good time) and any one of a range of different analog meters.

$7E - 126

$FF - 255

$7B - 123

$7F - 127

$6C - 108

$FE - 254

$7C - 124

$FB - 251

$E2 - 226

$EC - 236

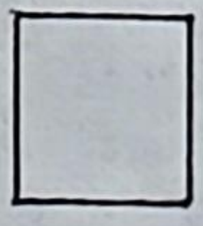
$62 - 98

$FC - 252

$E1 - 225

$A0 - 160

$61 - 97



$20 - 32

FIG 10. THE 16 QUARTER SQUARE GRAPHICS CHARACTERS

EXAMPLE 10

```
1000 REM ********************************
1001 REM *EXAMPLE OF SUBROUTINE CALL
1002 REM *TO DRAW GRAPH OF FUNCTION
1009 REM ********************************
1010 DEFFNP(X)=SIN(X/6.28)
1020 GOSUB24000
1030 END
2000 REM
3000 REM
4000 REM
5000 REM
6000 REM
24000 REM ********************************
24002 REM *SUBROUTINE TO DRAW A DOUBLE
24004 REM *DENSITY GRAPH OF THE FUNCTION
24006 REM *DEFINED BY - FNP(X)
24008 REM ********************************
24010 PRINT"□"
24012 FORX=32768TO33728STEP40:POKEX,101:NEXTX
24014 PRINT"■■■
```

Double Density Point Plot.

A routine to plot a point on the screen in double density format with a resolution of 80 x 50 on a 40 column machine. The coordinates 0,0 are located in the bottom left corner of the screen, the maximum value for the X coordinate is 79 and for the Y coordinate 49. Values outside these will set an error flag in location 856, this will have a value of 1 or 2 if out of range, 4 if a non quarter square character exists at these coordinates. The routine allows points to be deleted from the screen as well as added. A 0 in location 850 will put a point at the specified coordinates. A 1 in 850 will delete any quarter square displayed at the specified coordinates. The variable locations required by the routine are as follows:

848 .... X coordinate of point.
849 .... Y coordinate of point.
850 .... add or delete point, 0 = add, 1 = delete
856 .... error flag, 0 = OK, 1 or 2 = out of range, 4 = non plot character.

The routine is called from a Basic program with the command:

SYS(31371)

The following is an example of a Basic program using this routine:

```
 10 POKE848,25    :REM X COORDINATE = 25
 20 POKE849,12    :REM Y COORDINATE = 12
 30 POKE850,0     :REM ADD POINT TO DISPLAY
 40 SYS(31371)    :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE          LINE

1353  7A8B                ;**********************************
1354  7A8B                ;* SUBROUTINE TO PLOT POINTS
1355  7A8B                ;* IN DOUBLE DENSITY FORMAT
1356  7A8B                ;* ON THE PET.   GIVEN:
1357  7A8B                ;* X-COORD IN LOCATION $0350
1358  7A8B                ;* Y-COORD IN LOCATION $0351
1359  7A8B                ;* 0 IN LOCATION $0352 TO ADD
1360  7A8B                ;* 1 IN LOCATION $0352 TO DELETE
1361  7A8B                ;* ERROR FLAG IN LOCATION $0358
1362  7A8B                ;* 1 OR 2 = PLOT OUT OF RANGE
1363  7A8B                ;* 4 = NON-PLOTTABLE CHARACTER
1364  7A8B                ;*     ALREADY AT THESE COORDINATES
1365  7A8B                ;*     ON THE SCREEN
1366  7A8B                ;**********************************
1367  7A8B                ;
1368  7A8B                ;
1369  7A8B                ;
1370  7A8B  48            PPOINT PHA
1371  7A8C  98                   TYA
1372  7A8D  48                   PHA
1373  7A8E  8A                   TXA
1374  7A8F  48                   PHA
1375  7A90  A9 00                LDA #$0
1376  7A92  8D 58 03             STA ERROR
1377  7A95  8D 40 03             STA BINOFF
1378  7A98  85 01                STA T8+1
1379  7A9A  AD 51 03             LDA YCOORD
1380  7A9D  85 00                STA T8
1381  7A9F                ;
1382  7A9F                ;TEST IF Y-COORD IS >49
1383  7A9F  A5 00                LDA T8
1384  7AA1  C9 32                CMP #50
1385  7AA3  90 03                BCC YOK
1386  7AA5  EE 58 03             INC ERROR
1387  7AA8                ;
1388  7AA8                ;TEST IF XCOORD IS >79
1389  7AA8                ;
1390  7AA8  AD 50 03      YOK    LDA XCOORD
1391  7AAB  C9 50                CMP #80
1392  7AAD  90 03                BCC XOK
1393  7AAF  EE 58 03             INC ERROR
1394  7AB2                ;
1395  7AB2                ;RETURN IF OUT OF RANGE ERROR
1396  7AB2                ;
1397  7AB2  AD 58 03      XOK    LDA ERROR
1398  7AB5  F0 03                BEQ SORIG
1399  7AB7  4C 34 7B             JMP ENDP
1400  7ABA                ;
1401  7ABA                ;INVERT SCREEN FROM TOP TO BOTTOM
1402  7ABA                ;(Y-COORDINATE)
1403  7ABA                ;
1404  7ABA  A9 31         SORIG  LDA #49
1405  7ABC  38                   SEC
1406  7ABD  E5 00                SBC T8
1407  7ABF  85 00                STA T8
```

```
LINE# LOC    CODE            LINE

1408  7AC1                  ;
1409  7AC1                  ;SAVE BOTTOM BIT OF X-COORD
1410  7AC1                  ;IN BINOFF
1411  7AC1                  ;
1412  7AC1   4E 50 03             LSR XCOORD
1413  7AC4   2E 40 03             ROL BINOFF
1414  7AC7                  ;
1415  7AC7                  ;SAVE BOTTOM BIT OF Y-COORD
1416  7AC7                  ;IN BINOFF
1417  7AC7   46 00                LSR T8
1418  7AC9   2E 40 03             ROL BINOFF
1419  7ACC                  ;
1420  7ACC                  ;MULTIPLY T8 BY 40 AND
1421  7ACC                  ;ADD TO SCREEN ADDRESS
1422  7ACC                  ;
1423  7ACC   06 00                ASL T8
1424  7ACE   06 00                ASL T8
1425  7AD0   06 00                ASL T8
1426  7AD2   A5 00                LDA T8          ;SAVE IN A-REG
1427  7AD4   06 00                ASL T8
1428  7AD6   26 01                ROL T8+1
1429  7AD8   06 00                ASL T8
1430  7ADA   26 01                ROL T8+1
1431  7ADC   18                   CLC
1432  7ADD   65 00                ADC T8
1433  7ADF   85 00                STA T8
1434  7AE1   A5 01                LDA T8+1
1435  7AE3   69 80                ADC #$80        ;START OF SCREEN
1436  7AE5   85 01                STA T8+1
1437  7AE7                  ;
1438  7AE7                  ;EXPAND BINOFF
1439  7AE7                  ;
1440  7AE7   AE 40 03             LDX BINOFF
1441  7AEA   A9 01                LDA #1
1442  7AEC   8D 40 03             STA BINOFF
1443  7AEF   E0 00          EXP   CPX #0
1444  7AF1   F0 06                BEQ ENDEXP
1445  7AF3   0E 40 03             ASL BINOFF
1446  7AF6   CA                   DEX
1447  7AF7   90 F6                BCC EXP
1448  7AF9                  ;
1449  7AF9                  ;GET CHAR INTO A-REG
1450  7AF9                  ;
1451  7AF9   AC 50 03       ENDEXP LDY XCOORD
1452  7AFC   B1 00                LDA (T8),Y
1453  7AFE                  ;
1454  7AFE                  ;CHECK CHAR IS VALID GRAPHIC
1455  7AFE                  ;
1456  7AFE   A2 00                LDX #0
1457  7B00   DD 3A 7B       MOREC  CMP TABLE,X
1458  7B03   F0 0D                BEQ FOUND
1459  7B05   E8                   INX
1460  7B06   E0 10                CPX #16
1461  7B08   90 F6                BCC MOREC
1462  7B0A                  ;
```

```
LINE# LOC    CODE            LINE

 1463  7B0A                 ;CHAR IS NOT IN TABLE
 1464  7B0A                 ;
 1465  7B0A  A9 04                 LDA #4
 1466  7B0C  8D 58 03              STA ERROR
 1467  7B0F  4C 34 7B              JMP ENDP
 1468  7B12                 ;
 1469  7B12                 ;CHAR IS IN TABLE
 1470  7B12                 ;SO ERASE OR ADD
 1471  7B12                 ;
 1472  7B12  AD 52 03       FOUND  LDA AORD
 1473  7B15  D0 08                 BNE ERASPT
 1474  7B17                 ;
 1475  7B17                 ;ADD POINT TO SCREEN
 1476  7B17                 ;
 1477  7B17  8A             ADDPT  TXA
 1478  7B18  0D 40 03              ORA BINOFF
 1479  7B1B  18                    CLC
 1480  7B1C  AA                    TAX
 1481  7B1D  90 0D                 BCC NEWC
 1482  7B1F                 ;
 1483  7B1F                 ;ERASE POINT FROM SCREEN
 1484  7B1F                 ;
 1485  7B1F  AD 40 03       ERASPT LDA BINOFF
 1486  7B22  49 FF                 EOR #$FF
 1487  7B24  8D 40 03              STA BINOFF
 1488  7B27  8A                    TXA
 1489  7B28  2D 40 03              AND BINOFF
 1490  7B2B  AA                    TAX
 1491  7B2C                 ;
 1492  7B2C                 ;WRITE NEW GRAPHIC CHAR TO SCREEN
 1493  7B2C                 ;
 1494  7B2C  BD 3A 7B       NEWC   LDA TABLE,X
 1495  7B2F  AC 50 03              LDY XCOORD
 1496  7B32  91 00                 STA (T8),Y
 1497  7B34                 ;
 1498  7B34                 ;RETURN SUCCESSFULLY
 1499  7B34                 ;
 1500  7B34  68             ENDP   PLA
 1501  7B35  AA                    TAX
 1502  7B36  68                    PLA
 1503  7B37  A8                    TAY
 1504  7B38  68                    PLA
 1505  7B39  60                    RTS
 1506  7B3A                 ;
 1507  7B3A                 ;TABLE OF GRAPHICS CHARACTERS
 1508  7B3A                 ;
 1509  7B3A  20             TABLE  .BYTE $20,$7E,$7B,$61
 1509  7B3B  7E
 1509  7B3C  7B
 1509  7B3D  61
 1510  7B3E  7C                    .BYTE $7C,$E2,$FF,$EC
 1510  7B3F  E2
 1510  7B40  FF
 1510  7B41  EC
 1511  7B42  6C                    .BYTE $6C,$7F,$62,$FC
```

```
LINE# LOC    CODE          LINE

1511  7B43   7F
1511  7B44   62
1511  7B45   FC
1512  7B46   E1                    .BYTE $E1,$FB,$FE,$A0
1512  7B47   FB
1512  7B48   FE
1512  7B49   A0
```

Double Density Line Plot.

A routine to plot a line on the screen in double density format with a resolution of 80 x 50 on a 40 column machine. The line start and end points are defined as two sets of X and Y coordinates, the routine draws a line between these two points using quarter square characters. The coordinates 0,0 are located in the bottom left corner of the screen, the maximum value for the X coordinate is 79 and for the Y coordinate 49. Values outside these will set an error flag in location 857, a value of 1 indicates starting X coordinate is out of range, and 2 for end X out of range. A value of 3 in 857 indicates starting Y out of range and 4 for end Y out of range. Since this routine calls the point plot routine a 4 in 856 indicates that a non quarter square character was encountered during line drawing. This routine will not overwrite any characters already on the screen. Lines can be added, deleted or drawn as a dotted line on the screen depending on the value put in location 855. A 0 in location 855 will add a line joining the specified coordinates. A 1 in 855 will delete any double dinsity line displayed between the specified coordinates. A 2 in 855 will draw a dotted line between the specified coordinates. The variable locations required by the routine are as follows:

851 .... X coordinate of start of line.
852 .... Y coordinate of start of line.
853 .... X coordinate of end of line.
854 .... Y coordinate of end of line.
855 .... add or delete or dotted line, 0 = add, 1 = delete, 2 = dotted.
857 .... error flag, 0 = X start out, 2 = Y start out, 3 = X end out, 4 = Y end out.

The routine is called from a Basic program with the command:

SYS(31562)

The following is an example of a Basic program using this routine:

```
 10 POKE851,8     :REM X COORDINATE OF LINE START = 8
 20 POKE852,3     :REM Y COORDINATE OF LINE START = 3
 30 POKE853,42    :REM X COORDINATE OF LINE END = 42
 40 POKE854,38    :REM Y COORDINATE OF LINE END = 38
 50 POKE855,0     :REM ADD CONTINUOUS LINE TO DISPLAY
 60 SYS(31562)    :REM CALL ROUTINE
READY.
```

```
LINE# LOC    CODE            LINE

1514  7B4A                   ;**********************************
1515  7B4A                   ;*ROUTINE TO PLOT DOUBLE DENSITY
1516  7B4A                   ;*LINES ON SCREEN.  MOSTLY FOLLOWS
1517  7B4A                   ;*BASIC CODE
1518  7B4A                   ;*
1519  7B4A                   ;*X1 IN LOCATION $0353
1520  7B4A                   ;*Y1 IN LOCATION $0354
1521  7B4A                   ;*X2 IN LOCATION $0355
1522  7B4A                   ;*Y2 IN LOCATION $0356
1523  7B4A                   ;*
1524  7B4A                   ;*0 IN LOCATION $0357 TO ADD LINE
1525  7B4A                   ;*1 IN LOCATION $0357 TO REMOVE LINE
1526  7B4A                   ;*2 IN LOCATION $0357 TO DRAW DOTTED LINE
1527  7B4A                   ;*
1528  7B4A                   ;*ERROR FLAG IN LOCATION $0359
1529  7B4A                   ;*1 = X1 OUT OF RANGE
1530  7B4A                   ;*2 = X2 OUT OF RANGE
1531  7B4A                   ;*3 = Y1 OUT OF RANGE
1532  7B4A                   ;*4 = Y2 OUT OF RANGE
1533  7B4A                   ;*
1534  7B4A                   ;*DOES NOT WRITE OVER TEXT ALREADY
1535  7B4A                   ;*ON THE SCREEN
1536  7B4A                   ;*
1537  7B4A                   ;*CALLS PPLOT
1538  7B4A                   ;*
1539  7B4A                   ;**********************************
1540  7B4A                   ;
1541  7B4A  48               LPLOT  PHA
1542  7B4B  98                      TYA
1543  7B4C  48                      PHA
1544  7B4D  8A                      TXA
1545  7B4E  48                      PHA
1546  7B4F  A9 FF                   LDA #$FF
1547  7B51  8D 52 03                STA AORD
1548  7B54                   ;
1549  7B54                   ;CHECK IF COORDINATES IN BOUNDS
1550  7B54                   ;
1551  7B54  AD 53 03         S2030  LDA X1
1552  7B57  C9 50                   CMP #80
1553  7B59  90 05                   BCC TESTX2
1554  7B5B  A0 01                   LDY #1
1555  7B5D  4C A3 7C                JMP ER
1556  7B60  AD 55 03         TESTX2 LDA X2
1557  7B63  C9 50                   CMP #80
1558  7B65  90 05                   BCC TESTY1
1559  7B67  A0 02                   LDY #2
1560  7B69  4C A3 7C                JMP ER
1561  7B6C  AD 54 03         TESTY1 LDA Y1
1562  7B6F  C9 32                   CMP #50
1563  7B71  90 05                   BCC TESTY2
1564  7B73  A0 03                   LDY #3
1565  7B75  4C A3 7C                JMP ER
1566  7B78  AD 56 03         TESTY2 LDA Y2
1567  7B7B  C9 32                   CMP #50
1568  7B7D  90 05                   BCC S2100
```

```
LINE# LOC    CODE              LINE

 1569  7B7F   A0 04                    LDY #4
 1570  7B81   4C A3 7C                 JMP ER
 1571  7B84   AD 55 03     S2100       LDA X2          ;XD=X2-X1
 1572  7B87   38                       SEC
 1573  7B88   ED 53 03                 SBC X1
 1574  7B8B   8D 41 03                 STA XD
 1575  7B8E   AD 56 03     S2110       LDA Y2          ;YD=Y2-Y1
 1576  7B91   38                       SEC
 1577  7B92   ED 54 03                 SBC Y1
 1578  7B95   8D 42 03                 STA YD
 1579  7B98                ;
 1580  7B98                ;NEAREST DIAGONAL
 1581  7B98                ;
 1582  7B98   A9 01        S2130       LDA #1
 1583  7B9A   8D 43 03                 STA A0          ;A0=1
 1584  7B9D   8D 44 30                 STA A1          ;A1=1
 1585  7BA0   AD 42 03     S2140       LDA YD
 1586  7BA3   10 0A                    BPL S2150       ;IF YD<0 THEN 2150
 1587  7BA5   A0 FF                    LDY #$FF        ;A0=-1
 1588  7BA7   8C 43 03                 STY A0
 1589  7BAA   49 FF        S2170A      EOR #$FF          ;YE=ABS(YD)
 1590  7BAC   18                       CLC
 1591  7BAD   69 01                    ADC #1
 1592  7BAF   8D 45 03     S2150       STA YE
 1593  7BB2   AD 41 03                 LDA XD          ;IF XD<0 THEN 2170
 1594  7BB5   10 0A                    BPL S2170C
 1595  7BB7   A0 FF                    LDY #$FF        ;A1=-1
 1596  7BB9   8C 44 30                 STY A1
 1597  7BBC   49 FF        S2170B      EOR #$FF          ;XE=ABS(XD)
 1598  7BBE   18                       CLC
 1599  7BBF   69 01                    ADC #1
 1600  7BC1   8D 46 03     S2170C      STA XE
 1601  7BC4                ;
 1602  7BC4                ;NEAREST HORIZONTAL/VERTICAL
 1603  7BC4                ;
 1604  7BC4   AD 46 03                 LDA XE          ;D1=XE-YE
 1605  7BC7   38                       SEC
 1606  7BC8   ED 45 03                 SBC YE
 1607  7BCB   8D 47 03                 STA D1
 1608  7BCE   10 23        S2180       BPL S2220         ;IF D1>=0 THEN 2220
 1609  7BD0   A9 FF        S2190       LDA #$FF          ;S0=-1
 1610  7BD2   8D 48 03                 STA S0
 1611  7BD5   A9 00                    LDA #0          ;S1=0
 1612  7BD7   8D 49 03                 STA S1
 1613  7BDA   AD 45 03                 LDA YE          ;LG=YE
 1614  7BDD   8D 4A 03                 STA LG
 1615  7BE0   AD 46 03                 LDA XE          ;SH=XE
 1616  7BE3   8D 4B 03                 STA SH
 1617  7BE6   AD 42 03     S2200       LDA YD
 1618  7BE9   30 05                    BMI S2210       ;IF YD>=0 THEN S0=1
 1619  7BEB   A9 01                    LDA #1
 1620  7BED   8D 48 03                 STA S0
 1621  7BF0   4C 13 7C     S2210       JMP S2250         ;GOTO 2250
 1622  7BF3   A9 00        S2220       LDA #0            ;S0=0
 1623  7BF5   8D 48 03                 STA S0
```

mwk

```
LINE# LOC    CODE          LINE

1624  7BF8   A9 FF            LDA #$FF        ;S1=-1
1625  7BFA   8D 49 03         STA S1
1626  7BFD   AD 46 03         LDA XE          ;LG=XE
1627  7C00   8D 4A 03         STA LG
1628  7C03   AD 45 03         LDA YE          ;SH=YE
1629  7C06   8D 4B 03         STA SH
1630  7C09   AD 41 03  S2230  LDA XD           ;IF XD>=0 THEN S1=1
1631  7C0C   30 05            BMI S2250
1632  7C0E   A9 01            LDA #1
1633  7C10   8D 49 03         STA S1
1634  7C13                    ;
1635  7C13                    ;SET UP
1636  7C13                    ;
1637  7C13   AD 4A 03  S2250  LDA LG           ;TT=LG
1638  7C16   8D 4C 03         STA TT
1639  7C19   38               SEC             ;UD=LG-SH
1640  7C1A   ED 4B 03         SBC SH
1641  7C1D   8D 4E 03         STA UD
1642  7C20   AD 4B 03         LDA SH          ;TS=SH
1643  7C23   8D 4D 03         STA TS
1644  7C26   4E 4A 03         LSR LG          ;LG/2
1645  7C29   AD 4B 03         LDA SH          ;CT=SH-LG(/2)
1646  7C2C   38               SEC
1647  7C2D   ED 4A 03         SBC LG
1648  7C30   8D 4F 03         STA CT
1649  7C33                    ;
1650  7C33                    ;WHILE MORE POINTS DO
1651  7C33                    ;
1652  7C33   AD 53 03  S2270  LDA X1
1653  7C36   8D 50 03         STA XCOORD
1654  7C39   AD 54 03         LDA Y1
1655  7C3C   8D 51 03         STA YCOORD
1656  7C3F                    ;
1657  7C3F                    ;LOOK AT ARD TO ADD/REMOVE/DOTTED
1658  7C3F                    ;
1659  7C3F   AD 57 03  S2255  LDA ARD
1660  7C42   D0 04            BNE S2255A
1661  7C44   A9 00            LDA #0          ;ADD LINE
1662  7C46   F0 0D            BEQ DOPNT
1663  7C48   C9 01     S2255A CMP #1
1664  7C4A   D0 04            BNE S2255B
1665  7C4C   A9 01            LDA #1          ;REMOVE LINE
1666  7C4E   D0 05            BNE DOPNT
1667  7C50   AD 52 03  S2255B LDA AORD
1668  7C53   49 FF            EOR #$FF        ;ALTERNATELY ADD/REMOVE
1669  7C55   8D 52 03  DOPNT  STA AORD
1670  7C58                    ;
1671  7C58                    ;CALL POINT PLOT ROUTINE
1672  7C58                    ;
1673  7C58   20 8B 7A          JSR PPOINT
1674  7C5B   AD 4F 03  S2280  LDA CT           ;IF CT>=0 THEN 2320
1675  7C5E   10 1E            BPL S2320
1676  7C60   18        S2290  CLC              ;CT=CT+TS
1677  7C61   6D 4D 03         ADC TS
1678  7C64   8D 4F 03         STA CT
```

```
LINE# LOC    CODE            LINE

1679  7C67   AD 53 03              LDA X1          ;X1=X1+S1
1680  7C6A   18                    CLC
1681  7C6B   6D 49 03              ADC S1
1682  7C6E   8D 53 03              STA X1
1683  7C71   AD 54 03              LDA Y1          ;Y1=Y1+S0
1684  7C74   18                    CLC
1685  7C75   6D 48 03              ADC S0
1686  7C78   8D 54 03              STA Y1
1687  7C7B   4C 99 7C     S2310    JMP S2360        ;GOTO 2360
1688  7C7E   38           S2320    SEC              ;CT=CT-UD
1689  7C7F   ED 4E 03              SBC UD
1690  7C82   8D 4F 03              STA CT
1691  7C85   AD 53 03              LDA X1          ;X1=X1+A1
1692  7C88   18                    CLC
1693  7C89   6D 44 30              ADC A1
1694  7C8C   8D 53 03              STA X1
1695  7C8F   AD 54 03              LDA Y1          ;Y1=Y1+A0
1696  7C92   18                    CLC
1697  7C93   6D 43 03              ADC A0
1698  7C96   8D 54 03              STA Y1
1699  7C99   CE 4C 03     S2360    DEC TT           ;TT=TT-1
1700  7C9C   30 03        S2370    BMI RTN          ;IF TT<=0 THEN RETURN
1701  7C9E   4C 33 7C     S2380    JMP S2270        ;GOTO 2270
1702  7CA1   A0 00        RTN      LDY #0           ;ZERO ERROR
1703  7CA3   8C 59 03     ER       STY ERRORL       ;ERROR FLAG
1704  7CA6   68                    PLA
1705  7CA7   AA                    TAX
1706  7CA8   68                    PLA
1707  7CA9   A8                    TAY
1708  7CAA   68                    PLA
1709  7CAB   60                    RTS
```

EXAMPLE II

```
10 REM ********************************
20 REM *PROGRAM TO DRAW LINES ON SCREEN
30 REM *USING DOUBLE DENSITY MACHINE
40 REM *CODE PLOT SUBROUTINE - SYS(31371)
50 REM ********************************
60 REM
70 REM
80 PRINT"□";
90 PRINT"■";
100 INPUTX1,Y1,X2,Y2
110 GOSUB140
120 PRINT"■                              "
130 GOTO90
140 REM
150 REM **CHECK COORDINATES IN BOUND**
160 REM
170 IF(X1>=0ANDX1<=79)AND(X2>=0ANDX2<=79)THEN200
180 ER$="X OUT OF RANGE"
190 RETURN
200 IF(Y1>=0ANDY1<=49)AND(Y2>=0ANDY2<=49)THEN230
210 ER$="Y OUT OF RANGE"
220 RETURN
230 ER$=""
240 XD=X2-X1
250 YD=Y2-Y1
260 REM  **NEAREST DIAGONAL**
270 A0=1:A1=1
280 IFYD<0THENA0=-1
290 IFXD<0THENA1=-1
300 REM  **NEAREST HORIZ/VERT**
310 XE=ABS(XD):YE=ABS(YD):D1=XE-YE
320 IFD1>=0THEN360
330 S0=-1:S1=0:LG=YE:SH=XE
340 IFYD>=0THENS0=1
350 GOTO380
360 S0=0:S1=-1:LG=XE:SH=YE
370 IFXD>=0THENS1=1
380 REM  **SET UP**
390 TT=LG:TS=SH:UD=LG-SH:CT=SH-LG/2
400 D=0
410 REM  **WHILE MORE POINTS DO**
420 POKE848,X1:POKE849,Y1:POKE850,D:SYS(31371)
430 IFCT>=0THEN460
440 CT=CT+TS:X1=X1+S1:Y1=Y1+S0
450 GOTO470
460 CT=CT-UD:X1=X1+A1:Y1=Y1+A0
470 TT=TT-1
480 IFTT<=0THENRETURN
490 GOTO420
READY.
```

EXAMPLE 12

```
10 REM ************************************
20 REM *PROGRAM TO DRAW A CIRCLE OF POINTS
30 REM *USING DOUBLE DENSITY PLOT
40 REM ************************************
50 CX=40:CY=25:REM X/Y CENTRE COORDINATES
60 OX=1.2:OY=1:REM ELIPTICAL OFFSETS (CIRCLE OR ELIPSE?)
70 PRINT"□";
80 A=2*π
90 N=12:REM NUMBER OF PLOT POINTS
100 R=10:REM CIRCLE RADIUS
110 INC=(A-0)/N
120 FORI=0TOASTEPINC
130 X=R*SIN(I):X=INT(X*OX+CX+.499)
140 Y=R*COS(I):Y=INT(Y*OY+CY+.499)
150 POKE848,X:POKE849,Y:POKE850,D:SYS(31371)
160 NEXTI
170 PRINT"■"
180 END
READY.
```

EXAMPLE 13

```
10 REM ************************************
20 REM *PROGRAM TO SIMULATE A SATILITE ORBITING
30 REM *A PLANET USING DOUBLE DENSITY PLOT
40 REM ************************************
50 PRINT"□";
60 REM *DRAW PLANET AT CENTRE
70 POKE33268,85:POKE33269,73:POKE33308,74:POKE33309,75
80 CX=40:CY=25:REM X/Y COORDINATES
90 OX=1.2:OY=.65:REM ELIPTICAL OFFSETS
100 B=2*π
110 N=50:REM NUMBER OF PLOT POINTS
120 R=30:REM CIRCLE RADIUS
130 INC=(B-0)/N
140 FORI=0TOBSTEPINC
150 X=R*SIN(I):X=INT(X*OX+CX+.499)
160 Y=R*COS(I):Y=INT(Y*OY+CY+.499)
170 POKE848,X0:POKE849,Y0:POKE850,1:SYS(31371)
180 POKE848,X:POKE849,Y:POKE850,0:SYS(31371)
190 X0=X:Y0=Y
200 NEXTI
210 PRINT"■"
220 GOTO80
READY.
```

EXAMPLE 14

```
10 REM ***************************************
20 REM *PROGRAM TO DRAW A GRAPH BY INTERPOLATING A SET OF
30 REM *POINTS STORED AS DATA STATEMENTS IN LINE 170
40 REM *GRAPH PARAMETERS ARE IN LINES 210 AND 260
50 REM *POSITION DEFINED BY VALUES IN XL,XR AND YT,YB, DDENSITY
60 REM *MIN DIMENSION=DN, MAX =DX, SEPERATION OF POINTS =SP
70 REM ***************************************
80 PRINT"□";
90 DIMAX(12),AY(12)
100 FORI=1TO12
110 READAX(I)
120 READAY(I)
130 NEXTI
140 REM
150 REM *DATA STORED AS 'X' COORDINATE AND 'Y' COORDINATE
160 REM
170 DATA 1,10,2,20,3,30,4,50,5,40,6,30,7,50,8,10,9,20,10,50,11,30,12,10
180 REM
190 REM *MIN DIMENSION = 1, MAX =12, SEPERATION = 1
200 REM
210 DN=1:DX=12:SP=1
220 REM
230 REM *POSITION AND DIMENSIONS OF GRAPH ON SCREEN
240 REM *BOTTOM LEFT SCREEN IS COORDINATE 0,0
250 REM *HORIZONTAL - RIGHT, LEFT COLUMN. VERTICAL - TOP, BOTTOM LINE
260 XL=70:XR=10:YT=40:YB=10
270 REM
280 BL=INT((50-YT)/2)-2
290 BW=INT((XL-XR)/2)+1
300 BH=INT((YT-YB)/2)+4
310 BC=INT(XR/2)-1
320 REM
330 REM *DRAW BORDER AROUND GRAPH DISPLAY AREA.
340 REM
350 POKE 86,BC:POKE87,BL:POKE88,BW:POKE89,BH:SYS(30090)
360 REM
370 REM *CALCULATE SCALE FACTORS FOR GIVEN SCREEN SIZE AND POSITION
380 REM
390 Y1=-10000000
400 Y2=10000000
410 X1=Y1:X2=Y2
420 FORI=DN TO DX STEP SP
430 IFY1<AY(I)THENY1=AY(I)
440 IFY2>AY(I)THENY2=AY(I)
450 IFX1<AX(I)THENX1=AX(I)
460 IFX2>AX(I)THENX2=AX(I)
470 NEXTI
480 REM
490 REM *CONVERT SCALING FACTORS INTO POINT INCREMENTS
500 REM
510 A=(X1-X2)/(XL-XR)
520 B=(Y1-Y2)/(YT-YB)
530 REM
```

```
540 REM *PLOT GRAPH
550 REM
560 FORI=DN TO DX STEP SP
570 X=(XR+(AX(I)-X2)/A):Y=((AY(I)-Y2)/B+YB)
580 POKE848,X:POKE849,Y:POKE850,0:SYS(31371)
590 Q=I+SP
600 IFQ>DXTHEN670
610 FORJ=AX(I)TOAX(Q)STEP.03
620 Y3=((AY(Q)-AY(I))/(AX(Q)-AX(I)))*(J-AX(I))+AY(I)
630 X=INT(XR+(J-X2)/A):Y=INT((Y3-Y2)/B+YB)
640 POKE848,X:POKE849,Y:POKE850,0:SYS(31371)
650 NEXTJ
660 NEXTI
670 END
READY.
```

EXAMPLE 15

```
10 REM ************************************
20 REM CLOCK DEMONSTRATION USING DDENSITY LINE PLOTTING
30 REM ************************************
40 REM
50 POKE 48,00:POKE 49,112:REM LIMIT STRINGS TO $7000
60 POKE 52,00:POKE 53,112:REM LIMIT MEMORY TO $7000
70 POKE59459,255:REM PORT OUTPUT
80 POKE59471,255:REM ALL OUTPUTS HIGH
90 X1=851  :REM CONSTANTS
100 Y1=852
110 X2=853
120 Y2=854
130 D=855
140 REM DATA AREAS/TEXT STRINGS
150 DATA MONDAY,TUESDAY,WEDNESDAY
160 DATA THURSDAY,FRIDAY,SATURDAY,SUNDAY
170 DATA JANUARY,31,FEBRUARY,28
180 DATA MARCH,31,APRIL,30
190 DATA MAY,31,JUNE,30
200 DATA JULY,31,AUGUST,31
210 DATA SEPTEMBER,30,OCTOBER,31
220 DATA NOVEMBER,30,DECEMBER,31
230 REM SET UP
240 DIM ND$(7),NM$(12),DM(12),AD(7)
250 REM DAYS
260 FOR I=1TO7
270 READ ND$(I)
280 NEXT I
290 REM DAY/MONTH NAME
300 FOR I=1 TO 12
310 READ NM$(I),DM(I)
320 NEXT I
330 REM FORMAT
340 DW$="■■■■■■■■■■■■■"
350 AC$="■■■■■■■■■■■■"
360 B$="                    "
370 PI=3.14159265:CF=0:AT=0
380 REM SET DATE/TIME
390 GOSUB 1140
```

```
400 REM SET ALARM
410 GOSUB 2160
420 REM CLOCKFACE
430 GOSUB 1460
440 REM DATE
450 GOSUB 1720
460 REM CLOCK LOOP
470 HF=1:GOSUB 1010
480 GOTO 640
490 REM ERASE HOUR
500 IF HF=0 THEN570
510 POKE X1,XB
520 POKE Y1,YB
530 POKE X2,HD
540 POKE Y2,HC
550 POKE D,1
560 SYS(31562)
570 REM ERASE MIN
580 POKE X1,XB
590 POKE Y1,YB
600 POKE X2,MD
610 POKE Y2,MC
620 POKE D,1
630 SYS(31562)
640 REM INIT
650 REM DISP MIN
660 POKE X1,XB
670 POKE Y1,YB
680 POKE X2,MB
690 POKE Y2,MA
700 POKE D,0
710 SYS(31562)
720 REM DISP HOUR
730 POKE X1,XB
740 POKE Y1,YB
750 POKE X2,HB
760 POKE Y2,HA
770 POKE D,0
780 SYS(31562)
790 REM NEW TO OLD
800 IF HF=0 THEN830
810 HC=HA
820 HD=HB
830 MC=MA
840 MD=MB
850 REM CORRECTION AT 4AM
860 IF HO=3 AND HR=4 AND CF=0 THEN GOSUB2080
870 REM IF AM-PM UPDATE DATE
880 IF HO=11 AND HR=12 THEN GOSUB1720
890 REM IF NEW DAY NEW DATE
900 IF HO=23 AND HR=0 THEN GOSUB1920:GOSUB1720
910 REM ALARM ON
920 AT=AT+1
930 IF HR=AH AND MN=AM AND AD(DN)=1 THEN GOSUB 2120
940 IF AT>CA THEN POKE59471,255:PRINT"■";B$;B$;B$;B$;
950 HO=HR
960 REM NEXT MIN
970 GOSUB 2360
980 HF=0:IF MN/10=INT(MN/10)THEN HF=1
990 GOSUB 1010
```

```
1000 GOTO490
1010 REM SUB TO CALC COORDS
1020 REM OF THE HANDS
1030 XB=40:YB=25
1040 REM HOURS HAND
1050 IF HF=0 THEN 1090
1060 DG=((HR*30)+(INT(MN/10+.1)*5))*(PI/180)
1070 HA=INT(COS(DG)*12+.4999)+YB
1080 HB=INT(SIN(DG)*15+.4999)+XB
1090 REM MINUTES HAND
1100 DG=(MN*6)*(PI/180)
1110 MA=INT(COS(DG)*16+.4999)+YB
1120 MB=INT(SIN(DG)*20+.4999)+XB
1130 RETURN
1140 REM SUB TO SET DATE & TIME
1150 INPUT "DATE: DD,MM,YYYY";DY,MH,YR
1160 IF YR>1970 AND YR<1999 THEN 1190
1170 PRINT"YEAR BETWEEN 1970 & 1999 PLEASE"
1180 GOTO1150
1190 IF MH>0 AND MH<13 THEN 1220
1200 PRINT"MONTH BETWEEN JAN. AND DEC. PLEASE"
1210 GOTO1150
1220 GOSUB 2040:REM LEAP DAY!
1230 DT=DM(MH)+LP
1240 IF DY>0 AND DY<=DT THEN1270
1250 PRINT"NO SUCH DAY IN MONTH"
1260 GOTO 1150
1270 INPUT"DAY OF WEEK (MONDAY - 1)";DN
1280 IF DN>0 AND DN<8 THEN 1310
1290 PRINT"SEVEN DAYS IN A WEEK"
1300 GOTO 1270
1310 REM 24 HOUR CLOCK
1320 INPUT"HOUR OF THE DAY (0-23)";HR
1330 IF HR>=0 AND HR<24 THEN 1350
1340 GOTO 1320
1350 INPUT"MINUTE (0-59)";MN
1360 IF MN>=0 AND MN<60 THEN 1390
1370 PRINT"TRY AGAIN"
1380 GOTO1350
1390 REM LOAD PET CLOCK TI$
1400 HR$=RIGHT$("00"+STR$(HR),2)
1410 IF HR<10 THEN HR$="0"+RIGHT$(STR$(HR),1)
1420 MN$=RIGHT$(STR$(MN),2)
1430 IF MN<10 THEN MN$="0"+RIGHT$(STR$(MN),1)
1440 TI$=HR$+MN$+"00"
1450 RETURN
1460 REM THE CLOCK FACE
1470 PRINT"■"
1480 PRINT
1490 PRINTAC$;"■■■■■■■■■12"
1500 PRINTAC$;"■■11■■■■■■◆■■■■■1"
1510 PRINTAC$;"■■■■◆■■■■■■■■■◆"
1520 PRINT
1530 PRINT"■■■■■■■■10";AC$;AC$;"2"
1540 PRINTAC$;"◆";AC$;"■■■■■■■◆"
1550 PRINT
1560 PRINT
1570 PRINT
1580 PRINT
1590 PRINT"■■■■■■■8◆";AC$;AC$;"■■◆3"
```

```
1600 PRINT
1610 PRINT
1620 PRINT
1630 PRINT
1640 PRINT"■■■
```

```
2200 INPUT"ALARM MESSAGE";MS$
2210 PRINT"DAYS ALARM ON (Y OR N)"
2220 FOR I=1 TO 7
2230 PRINTND$(I);:INPUT A$
2240 AD(I)=0:IF A$="Y" THEN AD(I)=1
2250 NEXT I
2260 RETURN
2270 REM TEST ROUTINE
2280 MN=MN+1
2290 IF MN>59 THEN 2310
2300 RETURN
2310 HR=HR+1:MN=0
2320 IF HR>23 THEN 2340
2330 RETURN
2340 HR=0
2350 RETURN
2360 REM WAIT FOR NEXT MINUTE
2370 MO=MN
2380 MN=VAL(MID$(TI$,3,2))
2390 PRINT"■■■■■■■■■";RIGHT$(TI$,2)
2400 IF MO=MN THEN 2370
2410 HR=VAL(LEFT$(TI$,2))
2420 RETURN
READY.
```

EXAMPLE 16

```
10 REM ************************************
20 REM *THIS PROGRAM DEMONSTRATES THE USE OF DDENSITY LINE
30 REM *PLOTTING ROUTINE TO DISPLAY DATA INPUT ON THE USER
40 REM *PORT, IN THE FORM OF ONE OF A RANGE OF DIFFERENT
50 REM *ANALOG PANEL METER EMULATION DISPLAYS. FOR DETAILS
60 REM *ON USE OF USER PORT SEE 'PET REVEALED'.
70 REM ************************************
80 REM LIMIT TOP OF MEMORY
90 POKE52,00:POKE53,112
100 POKE48,00:POKE49,112:CLR
110 MN=0:    REM MINIMUM INPUT VALUE
120 MX=255:  REM MAXIMUM INPUT VALUE
130 DG=90  : REM POINTER TRAVEL (DEGREES)
140 LT=38:   REM LENGTH OF POINTER
150 XC=39:   REM POINTER FULCRUM (X)
160 YC=4:    REM POINTER FULCRUM (Y)
170 XA=0:    REM CALCULATED POINTER END (X)
180 YA=0:    REM CALCULATED POINTER END (Y)
190 XB=MN:   REM OLD XA
200 YB=MN:   REM OLD YA
210 ER=3:    REM ERROR BAND
220 IN=0:    REM INPUT VALUE
230 SR=MN-ER-2                :  REM OLD IN
240 LX=INT((LT*4/3)+.499):  REM 4:3 ASPECT RATIO
250 LY=LT
260 OF=360-INT(DG/2+.499):  REM METER CENTRE ALWAYS UP
270 AG=0:    REM CALCULATED POINTER ANGLE
280 TS=0:    REM TEST ROUTINE (101 INHIBITS POINTER REMOVAL)
290 X1=851:  REM SOME CONSTANTS
300 Y1=852
```

```
310 X2=853
320 Y2=854
330 D=855
340 LN=31562
350 REM CHOOSE METER LAYOUT
360 PRINT"■■■WHICH DEMONSTRATION WOULD YOU LIKE?■■"
370 PRINT"  1 - FOR LARGE 100 DEGREE METER"
380 PRINT"    (ANY KEY TO RETURN HERE)
390 PRINT"  2 - FOR LARGE 270 DEGREE METER"
400 PRINT"    (ANY KEY TO RETURN HERE)
410 PRINT"  3 - FOR FOUR SMALL METERS"
420 PRINT"    (INPUT NEGATIVE NUMBER TO RETURN)"
430 PRINT"  4 - CHANGE DEFAULT SETTINGS"
440 PRINT"  5 - RESTORE DEFAULT SETTINGS"
450 INPUT QU
460 IF QU<0 THEN STOP
470 ON QU GOTO 1130,1490,2730,820,110
480 STOP
490 REM DRAW POINTER ACCORDING TO IN
500 IF IN<MN THEN IN=MN
510 IF IN>MX THEN IN=MX
520 REM HAS POINTER MOVED (MORE THAN +-ER)?
530 IF IN>SR-ER AND IN<SR+ER THEN RETURN
540 SR=IN
550 REM CALCULATE POINTER COORDINATES
560 AG=(OF+(DG*(IN/(MX-MN))))*.0174533
570 YA=INT(COS(AG)*LY+.499)+YC
580 IF YA<0 OR YA>255 THEN 3420
590 XA=INT(SIN(AG)*LX+.499)+XC
600 IF XA<0 OR XA>255 THEN 3420
610 REM REMOVE OLD POINTER
620 IF TS=101 THEN700
630 POKE X1,XC
640 POKE Y1,YC
650 POKE X2,XB
660 POKE Y2,YB
670 POKE D,1
680 SYS(LN)
690 REM DRAW NEW POINTER
700 POKE X1,XC
710 POKE Y1,YC
720 POKE X2,XA
730 POKE Y2,YA
740 POKE D,0
750 SYS(LN)
760 REM TRANSFER
770 XB=XA
780 YB=YA
790 REM ERROR CHECK
800 IF PEEK(857)<>0 THEN 3420
810 RETURN
820 REM THE ROUTINE TO ALTER THE DEFAULT SETTINGS
830 PRINT"■■■■";
840 INPUT "MINIMUM INPUT VALUE (0)";MN
850 INPUT "MAXIMUM INPUT VALUE (255)";MX
860 IF MX>MN THEN 890
870 PRINT"MAX <= MIN, PLEASE INPUT AGAIN"
880 GOTO 840
890 INPUT "METER DEFLECTION ANGLE (90)";DG
900 IF DG>10 AND DG<=360 THEN 930
```

```
910 PRINT"DEFLECTION BETWEEN 10 AND 360 PLEASE"
920 GOTO 890
930 INPUT "LENGTH OF POINTER (38)";LT
940 IF LT>8 AND LT<46 THEN 970
950 PRINT"INAPPROPRIATE LENGTH"
960 GOTO 930
970 INPUT"CENTRE OF NEEDLE X,Y (39,4)";XC,YC
980 IF XC>=0 AND XC<=79 THEN 1010
990 PRINT"X OUT OF RANGE"
1000 GOTO 970
1010 IF YC>=0 AND YC<=49 THEN 1040
1020 PRINT"Y OUT OF RANGE"
1030 GOTO 970
1040 INPUT "ERROR BAND (3)";ER
1050 IF ER>0 THEN 1080
1060 PRINT"POSITIVE ERROR BAND ONLY"
1070 GOTO 1040
1080 REM SET UP REMAINDER
1090 LX=INT((LT*4/3)+.499)
1100 LY=LT
1110 OF=360-INT(DG/2+.499)
1120 GOTO 350
1130 REM DEMONSTRATION OF LARGE 90 DEGREE METER
1140 REM FIRST SET UP SCALE, LEGEND ETC.
1150 PRINT"■■■WHICH SCALE DO YOU REQUIRE?■"
1160 PRINT"  1 - LINEAR 0 TO 2"
1170 PRINT"  2 - LINEAR 0 TO 5"
1180 PRINT"  3 - LINEAR 0 TO 10"
1190 PRINT"  4 - DECIBEL SCALE"
1200 PRINT"  5 - NO SCALE"
1210 INPUT SC
1220 PRINT"■■■WHAT LEGEND DO YOU REQUIRE?■"
1230 PRINT"  1 - VOLTS"
1240 PRINT"  2 - AMPS"
1250 PRINT"  3 - OHMS"
1260 PRINT"  4 - A.C. SYMBOL"
1270 PRINT"  5 - DB SYMBOL"
1280 PRINT"  6 - NO SYMBOL"
1290 INPUT LG
1300 REM PRINT OUT SCALE
1310 PRINT"■";
1320 IF SC<0 THEN SC=5
1330 ON SC GOSUB 1810,1920,2030,2140
1340 REM PRINT SYMBOL
1350 IF LG<0 THEN LG=6
1360 ON LG GOSUB 2250,2350,2460,2550,2620
1370 REM PRINT METER COVER
1380 PRINT"■■■
```

```
1510 DG=270
1520 LT=18
1530 XC=39
1540 YC=22
1550 LX=INT((LT*4/3)+.499)
1560 LY=LT
1570 OF=360-(INT(DG/2)+.499)
1580 PRINT"
1590 PRINT"                              3000"
1600 PRINT"                              ◆"
1610 PRINT"
1620 PRINT"      2000                                  4000"
1630 PRINT"           ◆                                ◆"
1640 PRINT
1650 PRINT"                         R.P.M.
1660 PRINT"
1670 PRINT"                              "
1680 PRINT"                              "
1690 PRINT"                              "
1700 PRINT"   1000◆                                    ◆5000"
1710 PRINT"
1720 PRINT"                ◆                          ◆"
1730 PRINT"              0                             6000";
1740 REM METER READS FROM USER PORT
1750 POKE 59459,0:REM ALL LINES INPUTS
1760 GET A$:IF A$<>"" THEN 1760
1770 IN=PEEK(59471)
1780 GOSUB 490
1790 GET A$:IF A$="" THEN 1770
1800 GOTO 110
1810 REM LINEAR 0-2
1820 PRINT"
1830 PRINT"                    0.8   1.0   1.2"
1840 PRINT"              0.6   ◆     ◆     ◆     1.4"
1850 PRINT"        0.4   ◆                       ◆   1.6"
1860 PRINT"           ◆                                ◆"
1870 PRINT"   0.2                                            1.8"
1880 PRINT"     ◆                                            ◆"
1890 PRINT"0                                                      2"
1900 PRINT"                                                       ◆"
1910 RETURN
1920 REM LINEAR 0-5
1930 PRINT"
1940 PRINT"                         2           3"
1950 PRINT"                         ◆     ◆     ◆   "
1960 PRINT"           1     ◆                       ◆   4"
1970 PRINT"           ◆                                ◆"
1980 PRINT
1990 PRINT"     ◆                                            ◆"
2000 PRINT"0                                                      5"
2010 PRINT"                                                       ◆"
2020 RETURN
2030 REM LINEAR 0-10
2040 PRINT"
2050 PRINT"                               4     5     6"
2060 PRINT"                    3          ◆     ◆     ◆     7"
2070 PRINT"           2        ◆                             ◆   8"
2080 PRINT"           ◆                                      ◆"
2090 PRINT"     1                                                 9"
2100 PRINT"     ◆                                                 ◆"
```

```
2110 PRINT"0                                                    10"
2120 PRINT"▒▒▒
```

```
2710 PRINTR$;" ●● ● ●●●● "
2720 RETURN
2730 REM FOUR 75 DEGREE METERS
2740 REM SET UP DIFFERENT PARAMETERS
2750 LT=14
2760 ER=15
2770 DG=75
2780 OF=360-(DG/2+.499)
2790 LX=INT((LT*4/3)+.499)
2800 LY=LT
2810 REM PRINT METER FACES
2820 PRINT"■"
2830 FOR I=1 TO 2
2840 PRINT"    ┌──────────────────┐    ┌──────────────────┐"
2850 PRINT"    │        ◆         │    │        ◆         │"
2860 PRINT"    │                  │    │                  │"
2870 PRINT"    ├◆                ◆┤    ├◆                ◆┤"
2880 FOR J=1 TO 5
2890 PRINT"    │                  │    │                  │"
2900 NEXT J
2910 FOR J=1 TO 2
2920 PRINT"    ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒    ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
2930 NEXT J
2940 PRINT"    └──────────────────┘    └──────────────────┘";
2950 IF I=1 THEN PRINT
2960 NEXT I
2970 S1=-50:S2=-50:S3=-50:S4=-50
2980 INPUT"■1,2,3,4";I1,I2,I3,I4
2990 IF I1<MN OR I2<MN OR I3<MN OR I4<MN THEN 110
3000 IN=I1:XC=22:YC=28
3010 XB=XP:YB=YP:SR=S1
3020 GOSUB 490
3030 XP=XB:YP=YB:S1=SR
3040 IN=I2:XC=57:YC=28
3050 XB=XQ:YB=YQ:SR=S2
3060 GOSUB 490
3070 XQ=XB:YQ=YB:S2=SR
3080 IN=I3:XC=22:YC=4
3090 XB=XR:YB=YR:SR=S3
3100 GOSUB 490
3110 XR=XB:YR=YB:S3=SR
3120 IN=I4:XC=57:YC=4
3130 XB=XS:YB=YS:SR=S4
3140 GOSUB 490
3150 XS=XB:YS=YB:S4=SR
3160 GOTO2980
3170 REM TEST ROUTINE
3180 REM CALLING THIS ROUTINE SHOWS THE
3190 REM EXTENT OF THE ARC DRAWN BY THE
3200 REM POINTER.
3210 TS=101
3220 ES=ER
3230 IF ES=0 THEN ES=1
3240 FOR IN=MN TO MX STEP ES
3250 GOSUB 490
3260 NEXT IN
3270 TS=0
3280 RETURN
3290 REM TEST FOUR METER PANEL
3300 LT=14
```

```
3310 ER=15
3320 DG=130
3330 OF=360-(DG/2+.499)
3340 LX=INT((LT*.75)+.499)
3350 LY=LT
3360 PRINT"□"
3370 XC=22:YC=28:GOSUB3170
3380 XC=57:YC=28:GOSUB3170
3390 XC=22:YC=4 :GOSUB3170
3400 XC=57:YC=4 :GOSUB3170
3410 STOP
3420 REM PRINT ERROR MESSAGE
3430 PRINT"□■POINTER OFF SCREEN"
3440 PRINT"XC= ";XC
3450 PRINT"YC= ";YC
3460 PRINT"XA= ";XA
3470 PRINT"YA= ";YA
3480 PRINT"ANGLE WAS ";AG
3490 PRINT"INPUT WAS ";IN
3500 PRINT"■PLEASE RETYPE OPTIONS"
3510 GOTO 840
READY.
```

Notes

6 MULTIPLE SCREEN PAGE DISPLAYS

Multiple screen page displays - 139
Calculate big screen memory address from line and column - 142
Large screen initialisation - 144
Clear large screen memory - 146
Display specified part of large screen memory - 148
Save current screen contents to specified memory location - 150
Display large screen contents by page number - 152
Save screen to large page memory by page number - 154
Example uses above routines in simple word processor - 156
Example demonstrates use of large screen page display - 159

```
4K ADDRESSABLE MEMORY SPACE LON THE PET,
ROM.     THIS CONTAINS THE OPE RATING SYST
DDRESSS 49152 TO 65536 WITH A GAP BETWE
```

COLUMN 50 LINE 2 D O

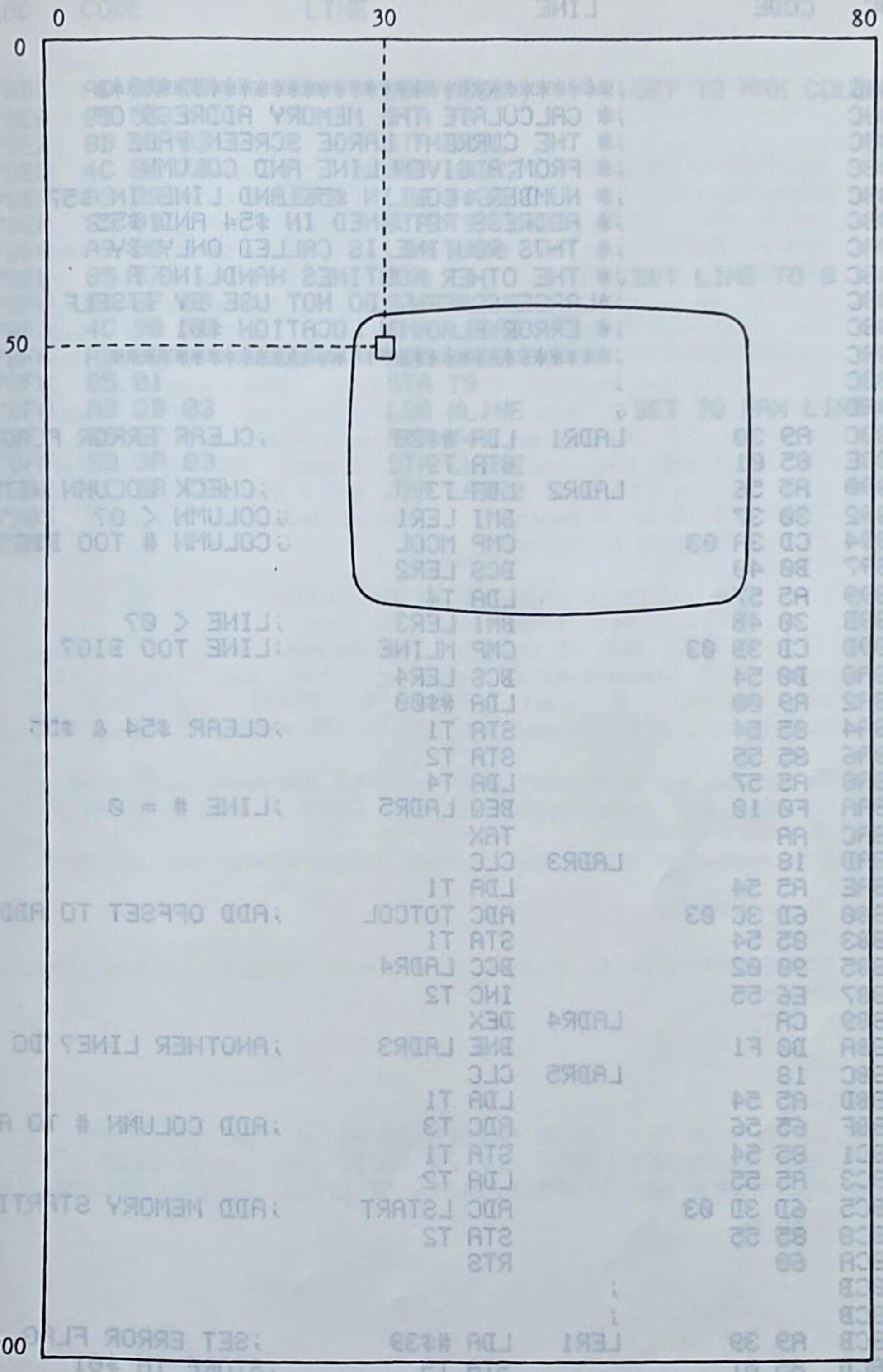
MULTIPLE SCREEN PACE DISPLAYS 6

There are a wide range of applications where the screen size of 40 columns and 25 lines is a limitation, principle amongst these is the requirement for the screen display to match the printed output, this would require an 80 column machine. One may need to display a list with more than 25 entries, but just one per screen line, this would require a screen with as many lines as there are entries to be displayed. Another problem is the time taken to display a screen mask of input prompts, messages, etc; standard information displayed by the program at that stage of the code and irrespective of the individual data enterd or displayed. It would be very convenient if several diffenent screen masks could be stored, once initialy set up, and then retreived and instantly displayed whenever required. All these problems confronting a programmer can be solved either completely or partialy with the routines in this section.

Creating a large multiple screen page display is simply a matter of enlarging the video memory, a 1000 bytes of memory will store one screen page on a 40 column machine. Of course the actual number of characters displayed can not be altered. The screen acts like a window onto the larger text area, the window being movable, enabling the entire large text area to be examined. To allow the screen to be located in the correct position over the large text area, a routine is required to calculate the starting memory location from the large screen line and column number. This routine is very similar to that used to calculate the starting address for a line or block displayed on the screen and like that routine is only used internaly by other routines and should not be used by itself. If any of the input values are out of range then an error flag is set in location 1, and the routine defaults to the maximum or minimum allowable. The size of the large screen is set by the initialisation routine, the number of columns width, the number of lines deep and the starting address of the memory area used to store the large screen. The initialisation routine sets the pointers to place the screen at the top left of the large text area and jumps to the clear screen routine which fills the whole of the large text area with space characters. The contents of any part of the large text area can be displayed, or the current screen contents saved onto it, by defining the required line and column number at which the top left corner of the screen is to be placed, and calling either the display or save routine. In some situations it may be more convenient to divide the large text area into screen pages and display or save it page by page, simply referencing page number rather than line and column, this is done by the last two routines in this

section.

The first example program EXAMPLE 17 shows how the large text area routines can be used to create a simple word processor. This program only implements a few of the functions which should be present on a word processor, however, the other functions can easily be added. Besides demonstrating the use of a large text area note also the use of a status line and of a simple repeat function. EXAMPLE 18 is an example program showing how individual pages of text can be saved and displayed.



Relationship between large screen text area and display.

```
LINE# LOC    CODE              LINE

0954  788C                    ;**********************************
0955  788C                    ;* CALCULATE THE MEMORY ADDRESS OF
0956  788C                    ;* THE CURRENT LARGE SCREEN PAGE
0957  788C                    ;* FROM A GIVEN LINE AND COLUMN
0958  788C                    ;* NUMBER, COL IN $56 AND LINE IN $57
0959  788C                    ;* ADDRESS RETURNED IN $54 AND $55
0960  788C                    ;* THIS ROUTINE IS CALLED ONLY BY
0961  788C                    ;* THE OTHER ROUTINES HANDLING A
0962  788C                    ;*LARGE SCREEN, DO NOT USE BY ITSELF
0963  788C                    ;* ERROR FLAG IN LOCATION $01
0964  788C                    ;**********************************
0965  788C                    ;
0966  788C                    ;
0967  788C  A9 30       LADR1  LDA #$30          ;CLEAR ERROR FLAG
0968  788E  85 01              STA T9
0969  7890  A5 56       LADR2  LDA T3            ;CHECK COLUMN WITHIN BO
0970  7892  30 37              BMI LER1          ;COLUMN < 0?
0971  7894  CD 3A 03           CMP MCOL          ;COLUMN # TOO BIG ?
0972  7897  B0 40              BCS LER2
0973  7899  A5 57              LDA T4
0974  789B  30 4B              BMI LER3          ;LINE < 0?
0975  789D  CD 3B 03           CMP MLINE         ;LINE TOO BIG?
0976  78A0  B0 54              BCS LER4
0977  78A2  A9 00              LDA #$00
0978  78A4  85 54              STA T1            ;CLEAR $54 & $55
0979  78A6  85 55              STA T2
0980  78A8  A5 57              LDA T4
0981  78AA  F0 10              BEQ LADR5         ;LINE # = 0
0982  78AC  AA                 TAX
0983  78AD  18          LADR3  CLC
0984  78AE  A5 54              LDA T1
0985  78B0  6D 3C 03           ADC TOTCOL        ;ADD OFFSET TO ADDRESS
0986  78B3  85 54              STA T1
0987  78B5  90 02              BCC LADR4
0988  78B7  E6 55              INC T2
0989  78B9  CA             LADR4  DEX
0990  78BA  D0 F1              BNE LADR3         ;ANOTHER LINE? DO AGAIN
0991  78BC  18          LADR5  CLC
0992  78BD  A5 54              LDA T1
0993  78BF  65 56              ADC T3            ;ADD COLUMN # TO ADDRESS
0994  78C1  85 54              STA T1
0995  78C3  A5 55              LDA T2
0996  78C5  6D 3D 03           ADC LSTART        ;ADD MEMORY STARTING ADD
0997  78C8  85 55              STA T2
0998  78CA  60                 RTS
0999  78CB                ;
1000  78CB                ;
1001  78CB  A9 39       LER1   LDA #$39          ;SET ERROR FLAG
1002  78CD  85 01              STA T9            ;STORE IN $01
1003  78CF  A9 00              LDA #$00
1004  78D1  85 56              STA T3            ;SET COLUMN TO 0
1005  78D3  8D 3E 03           STA CCOL
1006  78D6  4C 90 78           JMP LADR2         ;CALCULATE ADDRESS
1007  78D9  A9 39       LER2   LDA #$39
1008  78DB  85 01              STA T9
```

```
LINE# LOC    CODE          LINE

1009  78DD   AD 3A 03           LDA MCOL        ;SET TO MAX COLUMN #
1010  78E0   85 56              STA T3
1011  78E2   8D 3E 03           STA CCOL
1012  78E5   4C 90 78           JMP LADR2
1013  78E8   A9 39       LER3   LDA #$39
1014  78EA   85 01              STA T9
1015  78EC   A9 00              LDA #$00
1016  78EE   85 57              STA T4          ;SET LINE TO 0
1017  78F0   8D 3F 03           STA CLINE
1018  78F3   4C 90 78           JMP LADR2
1019  78F6   A9 39       LER4   LDA #$39
1020  78F8   85 01              STA T9
1021  78FA   AD 3B 03           LDA MLINE       ;SET TO MAX LINE #
1022  78FD   85 57              STA T4
1023  78FF   8D 3F 03           STA CLINE
1024  7902   4C 90 78           JMP LADR2
1025  7905                ;
1026  7905                ;
```

Large Screen Initialisation.

This initialisation routine must be run before any of the other routines associated with the large screen can be used. Its function is to initialise the variables and clear the memory area to be used for text storage. There are three variables which must be input to the routine they are; the number of columns wide and lines deep of the large screen and the most significant byte of the starting address of large screen memory. The top of memory must be lowered to the start of large screen memory, otherwise Basic will overwrite the text area, this is done in the first line of each example program. The starting address used will depend on the size of the large text area, each page will require 1K of memory, that used in EXAMPLE 17 will allow for 8 screen pages and in EXAMPLE 18 for 6 pages. The variable locations required by this routine area as follows:

826 .... large screen width in columns.
827 .... large screen depth in lines.
829 .... MSB of start of text storage memory address, convert to hexadecimal and then turn first two digits back into decimal - if start address is 20,480 then this in hexadecimal is $5000 and the MSB is $50 which is decimal 80.

To set the top of memory pointers POKE the same value used in location 829 into location 53, and POKE 0 into location 52.

The routine is called from a Basic program with the command:

SYS(30981)

The following is an example of a Basic program using this routine:

```
10 POKE 826,80   : REM SET LARGE SCREEN TO BE 80 COLUMNS WIDE
20 POKE 827,200  : REM SET LARGE SCREEN TO BE 200 LINES DEEP
30 POKE 829,80   : REM SET START ADDRESS OF LARGE SCREEN STORAGE TO HEX
```

```
LINE# LOC    CODE            LINE

1028  7905                  ;**********************************
1029  7905                  ;* LARGE SCREEN INITIALISATION AND
1030  7905                  ;* CLEAR
1031  7905                  ;* VARIABLES STORED BY POKE COMMANDS
1032  7905                  ;* $033A = NUMBER OF COLUMNS (EG. 40)
1033  7905                  ;* $033B = NUMBER OF LINES (EG. 200)
1034  7905                  ;* $033D = MSB OF DATA STORAGE START
1035  7905                  ;* ADDRESS (EG. IF $5000 THEN $50 OR
1036  7905                  ;* DECIMAL 80.)
1037  7905                  ;* ROUTINE CALLED BY - SYS(30981)
1038  7905                  ;**********************************
1039  7905                  ;
1040  7905                  ;
1041  7905  48              LINIT  PHA
1042  7906  98                     TYA
1043  7907  48                     PHA
1044  7908  8A                     TXA
1045  7909  48                     PHA
1046  790A  A9 00                  LDA #$00          ;INITIALISE VARIABLES
1047  790C  8D 3E 03               STA CCOL
1048  790F  8D 3F 03               STA CLINE
1049  7912  AD 3A 03               LDA MCOL
1050  7915  8D 3C 03               STA TOTCOL
1051  7918  E9 28                  SBC #$28
1052  791A  8D 3A 03               STA MCOL
1053  791D  AD 3B 03               LDA MLINE
1054  7920  E9 19                  SBC #$19
1055  7922  8D 3B 03               STA MLINE
1056  7925  4C 2D 79               JMP LCLR1         ;CLEAR LARGE SCREEN MEMORY
1057  7928                  ;
```

Clear Large Screen.

The entire contents of the large screen memory area is cleared by this routine, which sets all the locations in that area to decimal 32, the screen ASCII code for a space character. No variables are required by this routine.

The routine is called from a Basic program with the command:

SYS(31016)

The following is an example of a Basic program using this routine:

```
10 SYS(31016)  :REM CALL CLEAR SCREEN ROUTINE
```

```
LINE# LOC    CODE            LINE

 1059  7928                 ;**********************************
 1060  7928                 ;*CLEAR CONTENTS OF LARGE SCREEN
 1061  7928                 ;*MEMORY BY WRITING $20 (ASCII
 1062  7928                 ;*SPACE) INTO ALL LOCATIONS.
 1063  7928                 ;*ROUTINE CALLED BY - SYS(31016)
 1064  7928                 ;**********************************
 1065  7928                 ;
 1066  7928                 ;
 1067  7928  48            LCLR   PHA
 1068  7929  98                   TYA
 1069  792A  48                   PHA
 1070  792B  8A                   TXA
 1071  792C  48                   PHA
 1072  792D  A9 00         LCLR1  LDA #$00          ;SET UP START ADDRESS
 1073  792F  85 54                STA T1
 1074  7931  AD 3D 03             LDA LSTART
 1075  7934  85 55                STA T2
 1076  7936  A9 20         LCLR2  LDA #$20          ;ASCII SPACE CHARACTER
 1077  7938  A0 00                LDY #$00
 1078  793A  91 54         LCLR3  STA (T1),Y
 1079  793C  88                   DEY
 1080  793D  D0 FB                BNE LCLR3
 1081  793F  E6 55                INC T2
 1082  7941  A5 55                LDA T2
 1083  7943  C9 70                CMP #$70
 1084  7945  D0 EF                BNE LCLR2
 1085  7947  68                   PLA
 1086  7948  AA                   TAX
 1087  7949  68                   PLA
 1088  794A  A8                   TAY
 1089  794B  68                   PLA
 1090  794C  60                   RTS
 1091  794D
```

Large Screen Display.

Since the text area is larger than the actual screen size, this routine displays a screen sized section, the top left corner of which is at specified coordinates on the large screen text area. Thus if the large screen is 80 columns wide and 50 lines deep, it covers an area four times that of a 40 column screen. Specifying that the coordinates are column 0 and line 0 of the large screen, will display the left 40 columns and top 25 lines of the large screen text area. Similarly column 40 and line 0 will display the right hand 40 columns and top 25 lines ,or column 20 line 0 will display the central 40 columns of the top 25 lines. This procedure can be used to display any section of the large screen area providing it is within the limits of maximum column and line number. Thus line and column number should not be less than 0 nor should the column position be greater than the large screen width minus 40, or the line number greater than the large screen depth minus 25. This is because these display coordinates designate the top left corner of the screen, using larger values will result in an erroneous display. The variable locations required by this routine area as follows:

830 .... large screen column number where top left of display starts.

831 .... large screen line number where top left of display starts.

The routine is called from a Basic program with the command:

SYS(31053)

The following is an example of a Basic program using this routine:

```
10 POKE 830,30  : REM TOP LEFT OF DISPLAY = LARGE SCREEN COLUMN NUMBER
20 POKE 831,50  : REM TOP LEFT OF DISPLAY = LARGE SCREEN LINE NUMBER 3
30 SYS (31053)  : REM CAL LARGE SCREEN DISPLAY ROUTINE
```

```
LINE# LOC    CODE          LINE

1093  794D                ;**********************************
1094  794D                ;*DISPLAY PAGE OF TEXT POINTED TO
1095  794D                ;*BY POINTERS IN $033E (COL) AND
1096  794D                ;*$033F (LINE)
1097  794D                ;*ROUTINE CALLED BY - SYS(31053)
1098  794D                ;**********************************
1099  794D                ;
1100  794D                ;
1101  794D  48            LDISP  PHA
1102  794E  98                    TYA
1103  794F  48                    PHA
1104  7950  8A                    TXA
1105  7951  48                    PHA
1106  7952  AD 3E 03              LDA CCOL       ;TRANSFER COLUMN AND LINE
1107  7955  85 56                 STA T3         ;VARIABLES TO WORKING STORAGE
1108  7957  AD 3F 03              LDA CLINE
1109  795A  85 57                 STA T4
1110  795C  20 8C 78    LDISP0    JSR LADR1       ;CALCULATE MEMORY ADDRESS
1111  795F  A9 00                 LDA #$00
1112  7961  85 56                 STA T3
1113  7963  A9 80                 LDA #$80       ;SCREEN START ADDRESS
1114  7965  85 57                 STA T4
1115  7967  A2 17                 LDX #$17       ;MAX SCREEN LINE #
1116  7969  A0 27       LDISP1    LDY #$27        ;MAX SCREEN COLUMN #
1117  796B  B1 54       LDISP2    LDA (T1),Y
1118  796D  91 56                 STA (T3),Y     ;DISPLAY MEMORY ON SCREEN
1119  796F  88                    DEY
1120  7970  10 F9                 BPL LDISP2
1121  7972  18                    CLC
1122  7973  A5 54                 LDA T1
1123  7975  6D 3C 03              ADC TOTCOL     ;ADD OFFSET VALUE
1124  7978  85 54                 STA T1
1125  797A  90 02                 BCC LDISP3
1126  797C  E6 55                 INC T2
1127  797E  18          LDISP3    CLC
1128  797F  A5 56                 LDA T3
1129  7981  69 28                 ADC #$28       ;ADD SCREEN OFFSET
1130  7983  85 56                 STA T3
1131  7985  90 02                 BCC LDISP4
1132  7987  E6 57                 INC T4 ,
1133  7989  CA          LDISP4    DEX
1134  798A  D0 DD                 BNE LDISP1
1135  798C  68                    PLA
1136  798D  AA                    TAX
1137  798E  68                    PLA
1138  798F  A8                    TAY
1139  7990  68                    PLA
1140  7991  60                    RTS
1141  7992                ;
```

Large Screen Save.

This routine saves the contents of the screen and records it in the large screen memory area, in a position corresponding to the coordinates specified. The coordinates refer to the line and column number of the large screen and designate the position of the top left corner of the actual PET screen. The constraints on coordinate size are the same as that defined for large screen display. By careful use of this and the previous display routine one can enter and display data on the large screen as if it were a normal screen. Displays stored in the large screen memory can be saved and then retreived at a later date by storing that area of memory on tape or disk using the monitor program. The variable locations required by the program are as follows:

830 .... large screen column number where top left of display starts.

831 .... large screen line number where top left of display starts.

The routine is called from a Basic program with the command:

SYS(31122)

The following is an example of a Basic program using this routine:

```
10 POKE 830,30  : REM TOP LEFT OF DISPLAY = LARGE SCREEN COLUMN NUMBER
20 POKE 831,50  : REM TOP LEFT OF DISPLAY = LARGE SCREEN LINE NUMBER 3
30 SYS (31122)  : REM CAL LARGE SCREEN SAVE ROUTINE
```

```
LINE# LOC    CODE            LINE

 1143  7992                 ;**********************************
 1144  7992                 ;*SAVE CURRENT SCREEN CONTENTS INTO
 1145  7992                 ;*MEMORY AT THE CORRECT LOCATION
 1146  7992                 ;*DETERMINED BY THE CONTENTS OF
 1147  7992                 ;*$033E - COL # AND $033F - LINE #
 1148  7992                 ;*ROUTINE CALLED BY - SYS(31122)
 1149  7992                 ;**********************************
 1150  7992                 ;
 1151  7992                 ;
 1152  7992  48            LSAVE  PHA
 1153  7993  98                   TYA
 1154  7994  48                   PHA
 1155  7995  8A                   TXA
 1156  7996  48                   PHA
 1157  7997  AD 3E 03             LDA CCOL        ;TRANSFER COLUMN AND LINE
 1158  799A  85 56                STA T3          ;VARIABLES TO WORK AREA
 1159  799C  AD 3F 03             LDA CLINE
 1160  799F  85 57                STA T4
 1161  79A1  20 8C 78    LSAVE0   JSR LADR1        ;CALCULATE MEMORY ADDRESS
 1162  79A4  A9 00                LDA #$00
 1163  79A6  85 56                STA T3
 1164  79A8  A9 80                LDA #$80        ;SCREEN START ADDRESS
 1165  79AA  85 57                STA T4
 1166  79AC  A2 17                LDX #$17        ;MAX SCREEN LINE #
 1167  79AE  A0 27    LSAVE1      LDY #$27        ;MAX SCREEN COLUMN #
 1168  79B0  B1 56    LSAVE2      LDA (T3),Y
 1169  79B2  91 54                STA (T1),Y      ;TRANSFER SCREEN TO MEMORY
 1170  79B4  88                   DEY
 1171  79B5  10 F9                BPL LSAVE2
 1172  79B7  18                   CLC
 1173  79B8  A5 54                LDA T1
 1174  79BA  6D 3C 03             ADC TOTCOL      ;ADD MEMORY OFFSET
 1175  79BD  85 54                STA T1
 1176  79BF  90 02                BCC LSAVE3
 1177  79C1  E6 55                INC T2
 1178  79C3  18          LSAVE3   CLC
 1179  79C4  A5 56                LDA T3
 1180  79C6  69 28                ADC #$28        ;ADD SCREEN OFFSET
 1181  79C8  85 56                STA T3
 1182  79CA  90 02                BCC LSAVE4
 1183  79CC  E6 57                INC T4
 1184  79CE  CA          LSAVE4   DEX
 1185  79CF  D0 DD                BNE LSAVE1
 1186  79D1  68                   PLA
 1187  79D2  AA                   TAX
 1188  79D3  68                   PLA
 1189  79D4  A8                   TAY
 1190  79D5  68                   PLA
 1191  79D6  60                   RTS
 1192  79D7
```

Display Large Screen Page.

In some applications it is not practical to access a page of text from large screen memory using line and column number, it is easier to reference it by page number. This routine requires the large screen memory to be initialised with the initialisation routine so that it is 40 columns wide and up to six page lengths or 150 lines long. Any one of the six pages can be displayed by putting the required page number into location 88 and calling the routine. If fewer pages are required then the initialisation routine should be given the correct number of lines for the required number of pages. The variable locations required by the program are as follows:

88 .... page number to be displayed.

The routine is called from a Basic program with the command:

SYS(31191)

The following is an example of a Basic program using this routine:

```
10 POKE 88,3    : REM DISPLAY PAGE #3 OF LARGE SCREEN TEXT
20 SYS (31191) : REM CALL LARGE SCREEN PAGE DISPLAY ROUTINE
```

```
LINE# LOC    CODE          LINE

1194  79D7                ;**********************************
1195  79D7                ;*DISPLAY SCREEN PAGE USING MEMORY
1196  79D7                ;*SET UP FOR LARGE SCREEN SCROLL.
1197  79D7                ;*LARGE SCREEN MUST BE SET UP FOR
1198  79D7                ;*ONE PAGE WIDTH ONLY. PAGE # STORED
1199  79D7                ;*IN $58. MAXIMUM PAGE #=9 & DEPENDS ON
1200  79D7                ;*INITIALISATION USED IN LARGE SCREEN.
1201  79D7                ;*ROUTINE CALLED BY - SYS(31191)
1202  79D7                ;**********************************
1203  79D7                ;
1204  79D7                ;
1205  79D7  48            DPAGE  PHA
1206  79D8  98                   TYA
1207  79D9  48                   PHA
1208  79DA  8A                   TXA
1209  79DB  48                   PHA
1210  79DC  A9 00                LDA #$00          ;INITIALISE VARIABLES
1211  79DE  85 56                STA T3            ;COLUMN
1212  79E0  85 57                STA T4            ;LINE
1213  79E2  18                   CLC
1214  79E3  A6 58                LDX T5            ;PAGE NUMBER
1215  79E5  A5 57          DPAG1 LDA T4             ;LINE NUMBER
1216  79E7  69 18                ADC #$18
1217  79E9  85 57                STA T4
1218  79EB  CA                   DEX
1219  79EC  D0 F7                BNE DPAG1         ;DO AGAIN FOR NEXT PAGE
1220  79EE  4C 5C 79             JMP LDISP0        ;DISPLAY PAGE
1221  79F1                ;
```

Save Large Screen Page.

This routine will save the current screen contents into the large screen memory area at a position which depends on the specified page number. This routine requires the large screen memory to be initialised with the initialisation routine so that it is 40 columns wide and up to six page lengths or 150 lines long. Any one of the six pages can be saved by putting the required page number into location 88 and calling the routine. This page can be retreived again using the screen page display routine. If fewer pages are required then the initialisation routine should be given the correct number of lines for the required number of pages. The variable locations required by the program are as follows:

88 .... page number to be saved.

The routine is called from a Basic program with the command:

SYS(31217)

The following is an example of a Basic program using this routine:

```
10 POKE 88,3    : REM DISPLAY PAGE #3 OF LARGE SCREEN TEXT
20 SYS (31217) : REM CALL LARGE SCREEN PAGE SAVE ROUTINE
```

```
LINE# LOC   CODE          LINE

1223  79F1                ;**********************************
1224  79F1                ;*SAVE SCREEN PAGE TO MEMORY SET
1225  79F1                ;*UP FOR LARGE SCROLLED SCREEN.
1226  79F1                ;*PAGE # STORED IN $58.
1227  79F1                ;*ROUTINE CALLED BY - SYS(31217)
1228  79F1                ;**********************************
1229  79F1                ;
1230  79F1                ;
1231  79F1  48            SPAGE  PHA
1232  79F2  98                   TYA
1233  79F3  48                   PHA
1234  79F4  8A                   TXA
1235  79F5  48                   PHA
1236  79F6  A9 00                LDA #$00        ;INITIALISE VARIABLES
1237  79F8  85 56                STA T3          ;COLUMN
1238  79FA  85 57                STA T4          ;LINE
1239  79FC  18                   CLC
1240  79FD  A6 58                LDX T5          ;PAGE #
1241  79FF  A5 57          SPAG1 LDA T4           ;LINE  #
1242  7A01  69 18                ADC #$18
1243  7A03  85 57                STA T4
1244  7A05  CA                   DEX
1245  7A06  D0 F7                BNE SPAG1       ;DO AGAIN FOR NEXT PAGE
1246  7A08  4C A1 79             JMP LSAVE0      ;SAVE PAGE
1247  7A0B                ;
1248  7A0B                ;
```

EXAMPLE 17

```
10 REM ************************************
20 REM *DEMONSTRATION OF ROUTINES TO CONTROL LARGE SCREEN TEXT
30 REM *AREA. THE PROGRAM EMULATES SOME OF THE FUNCTIONS OF A
40 REM *SIMPLE WORD PROCESSOR. SINCE MOST OF THE CODE IS WRITTEN
50 REM *IN BASIC IT IS VERY SLOW.
60 REM ************************************
70 POKE52,0:POKE53,80
80 POKE48,0:POKE49,80:CLR
90 REM
100 REM INITIALISE POINTERS FOR LARGE SCREEN
110 REM
120 POKE826,82
130 POKE827,77
140 POKE829,80
150 SYS(30981)
160 LP=0:CP=0:CC=0:LC=0:C=0:P=0
170 REM
180 REM PUT CURSOR AT TOP LEFT AND SET UP CONTROL CHARACTERS
190 REM
200 PRINT"□"
210 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
220 POKE33760,3:POKE33762,4:POKE33764,9:POKE33766,15
230 GOSUB430
240 REM
250 REM GET CURSOR CONTROL INPUTS
260 REM
270 GETA$:IFA$=""THEN270
280 IFA$="■"THENGOSUB560
290 IFA$="■"THENGOSUB690
300 IFA$="■"THENGOSUB820
310 IFA$="□"THENGOSUB950
320 IFA$="■"THENGOSUB1080
330 REM
340 REM 'RVS' IS CONTROL KEY IF PRESSED THEN 'C' ON CONTROL LINE
350 REM IS REVERSED AND CONTROL OPTIONS ENABLED, THESE ARE:
360 REM I -INPUT DATA TO SCREEN. O -OUTPUT TO PRINTER
370 REM D -DELETE ENTIRE LARGE SCREEN CONTENTS
380 REM
390 IFA$="■"THENPOKE33760,131:GOSUB1170
400 POKE 151,255
410 GOTO230
420 REM
430 REM DISPLAY CONTROL LINE
440 REM
450 POKE86,0:POKE87,23:POKE88,38:POKE0,67:SYS(30029)
460 POKE84,24:POKE85,0:SYS(31243)
470 PRINT"          COLUMN      LINE          ";
480 POKE84,24:POKE85,14:SYS(31243)
490 PRINTCC;
500 POKE84,24:POKE85,25:SYS(31243)
510 PRINTLC;
520 RETURN
530 REM
```

```
540 REM CURSOR CONTROL SUBROUTINES
550 REM
560 REM CURSOR UP
570 REM
580 LC=LC+1:IFLC>23ORLC=23THENL=L+1
590 IFLC>75THENLC=75
600 IFL>50THENL=50
610 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
620 LP=LC
630 IFLC>22THENLP=22
640 POKE831,L
650 SYS(31053)
660 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
670 RETURN
680 REM
690 REM CURSOR LEFT
700 REM
710 CC=CC-1:IFCC>39ORCC=39THENC=C-1
720 IFCC<0THENCC=0
730 IFC<0THENC=0
740 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
750 CP=CC
760 IFCC>39ORCC=39THENCP=39
770 POKE830,C
780 SYS(31053)
790 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
800 RETURN
810 REM
820 REM CURSOR RIGHT
830 REM
840 CC=CC+1:IFCC>40ORCC=40THENC=C+1
850 IFCC>80THENCC=80
860 IFC>40THENC=40
870 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
880 CP=CC
890 IFCC>39ORCC=39THENCP=39
900 POKE830,C
910 SYS(31053)
920 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
930 RETURN
940 REM
950 REM CURSOR DOWN
960 REM
970 LC=LC-1:IFLC>22ORLC=22THENL=L-1
980 IFLC<0THENLC=0
990 IFL<0THENL=0
1000 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1010 LP=LC
1020 IFLC>22THENLP=22
1030 POKE831,L
1040 SYS(31053)
1050 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1060 RETURN
1070 REM
1080 REM CURSOR HOME
1090 REM
1100 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1110 CC=0:C=0:CP=0:POKE830,C
1120 LC=0:L=0:LP=0:POKE831,L
1130 SYS(31053)
```

```
1140 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1150 RETURN
1160 REM
1170 REM INPUT CONTROL FUNCTION KEY - PREES 'RVS' TO EXIT CONTROL MODE
1180 REM
1190 GOSUB430
1200 GETA$:IFA$=""THEN1200
1210 IFA$="D"THEN1270
1220 IFA$="I"THEN1410
1230 IFA$="O"THEN1750
1240 IFA$="■"THENPOKE33760,3:FORQ=1TO100:NEXT:RETURN
1250 GOTO1190
1260 REM
1270 REM PERFORM LARGE SCREEN ERASE FUNCTION - 'CONTROL' AND 'D'
1280 REM
1290 POKE33762,132
1300 POKE84,24:POKE85,0:SYS(31243)
1310 PRINT"■DELETE?■";
1320 GETA$:IFA$=""THEN1320
1330 IFA$<>"Y"THEN1370
1340 SYS(31016)
1350 SYS(31053)
1360 GOSUB1080
1370 GOSUB430
1380 POKE33760,3:POKE33762,4
1390 RETURN
1400 REM
1410 REM PERFORM DATA INPUT FUNCTION - 'CONTROL' AND 'I'
1420 REM
1430 POKE33764,137
1440 GOSUB430
1450 POKE151,255:REM REPEAT
1460 GETA$:IFA$=""THEN1460
1470 IFA$="■"THENGOSUB560:GOTO1440
1480 IFA$="■"THENGOSUB690:GOTO1440
1490 IFA$="■"THENGOSUB820:GOTO1440
1500 IFA$="■"THENGOSUB950:GOTO1440
1510 IFA$="■"THENGOSUB1080:GOTO1440
1520 REM
1530 REM PRESS 'RVS' TO EXIT INPUT MODE
1540 REM
1550 IFA$="■"THENPOKE33760,3:POKE33764,9:FORQ=1TO100:NEXT:RETURN
1560 POKE84,LP:POKE85,CP:SYS(31243)
1570 PRINTA$;
1580 SYS(31122)
1590 IFCC=79THEN1620
1600 GOSUB820
1610 GOTO1440
1620 CC=0:CP=0:C=0
1630 POKE830,C
1640 LC=LC+1:IFLC>22THENL=L+1
1650 IFLC>75THENLC=75
1660 IFL>50THENL=50
1670 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1680 LP=LC
1690 IFLC>22THENLP=22
1700 POKE831,L
1710 SYS(31053)
1720 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1:SYS(30510)
1730 GOTO1440
```

```
1740 REM
1750 REM OUTPUT CONTENTS OF LARGE SCREEN TO PRINTER - 'CONTROL' AND 'O'
1760 REM
1770 POKE33766,143
1780 OPEN4,4
1790 FORQ=20480TO20800STEP82
1800 FORX=QTOQ+79
1810 A=PEEK(X)
1820 B=A
1830 IFA<32THENB=A+64:IFA>63THENB=32
1840 A$=CHR$(B)
1850 PRINT#4,A$;
1860 NEXTX
1870 PRINT#4,CHR$(13);
1880 NEXTQ
1890 CLOSE4
1900 POKE33760,3:POKE33766,15
1910 RETURN
READY.
```

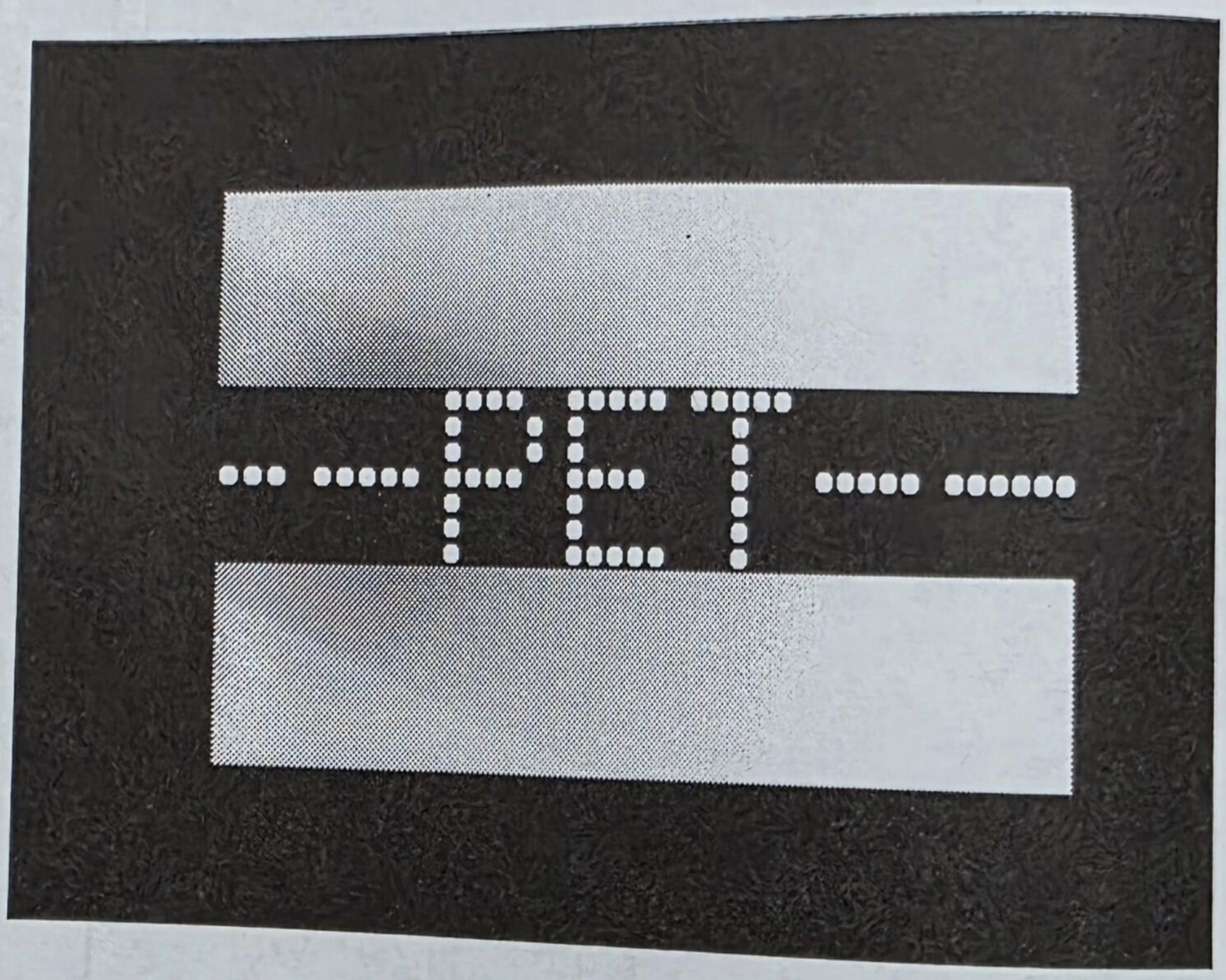
EXAMPLE 18

```
10 REM ***************************************
20 REM *DEMONSTRATION OF MULTIPLE SCREEN PAGE SAVE AND DISPLAY
30 REM *USING LARGE SCREEN ROUTINES. WILL SAVE UP TO SIX SEPERATE
40 REM *SCREEN PAGES.
50 REM ***************************************
60 POKE48,0:POKE49,88
70 POKE52,0:POKE53,88:CLR
80 REM
90 REM INITIALISE VARIABLE POINTERS FOR LARGE SCREEN
100 REM
110 POKE826,40
120 POKE827,150
130 POKE829,88
140 SYS(30981)
150 REM
160 REM
170 REM
180 REM CREATE SIX DEMONSTRATION SCREEN PAGES AND STORE FOR LATER
190 REM RECALL. ENTER ANY TEXT OF YOUR OWN AND THEN PRESS 'HOME'
200 REM TO SAVE THAT PAGE IN MEMORY.
210 REM
220 PRINT"□"
230 DATA"DEMO PAGE ONE","DEMO PAGE TWO","DEMO PAGE THREE"
240 DATA"DEMO PAGE FOUR","DEMO PAGE FIVE","DEMO PAGE SIX"
250 POKE84,5:POKE85,5:SYS(31243)
260 PRINT"■CREATE DEMONSTRATION PAGES AND"
270 POKE84,6:POKE85,12:SYS(31243)
280 PRINT"■STORE IN MEMORY"
290 POKE84,10:POKE85,4:SYS(31243)
300 PRINT"■PRESS 'HOME' TO CREATE NEXT PAGE■"
310 GETA$:IFA$=""THEN310
320 IFA$<>"■"THEN310
330 PRINT"□"
340 FORQ=1TO6
350 A=Q+48
```

```
360 POKE86,0:POKE87,0:POKE88,39:POKE89,23:SYS(30090)
370 POKE84,10:POKE85,13:SYS(31243)
380 READA$:PRINTA$
390 GETA$:IFA$=""THEN390
400 IFA$="■"THEN430
410 PRINTA$;
420 GOTO390
430 POKE88,Q:SYS(31217)
440 PRINT"□"
450 NEXTQ
460 REM
470 REM
480 REM
490 REM
500 REM DISPLAY ANY OF THE PREVIOUSLY CREATED SIX PAGES.
510 REM
520 POKE84,10:POKE85,5:SYS(31243)
530 PRINT"■DISPLAY DEMONSTRATION PAGES■"
540 POKE 86,0:POKE87,23:POKE88,40:POKE0,32:SYS(30029)
550 POKE 86,0:POKE87,24:POKE88,40:POKE0,32:SYS(30029)
560 POKE84,23:POKE85,0:SYS(31243)
570 PRINT"■PAGE # ";
580 INPUTA
590 IFA<1ORA>6THEN540
600 POKE88,A:SYS(31191)
610 GOTO540
READY.
```

Displaying and moving character blocks 'Macro characters' - 163
Routine displays 'macro character' from character table - 166
Example gives a moving signboard in large characters - 169
Example shows controled moving of 'macro characters' - 171

7 DISPLAYING AND MOVING 'MACRO CHARACTERS'

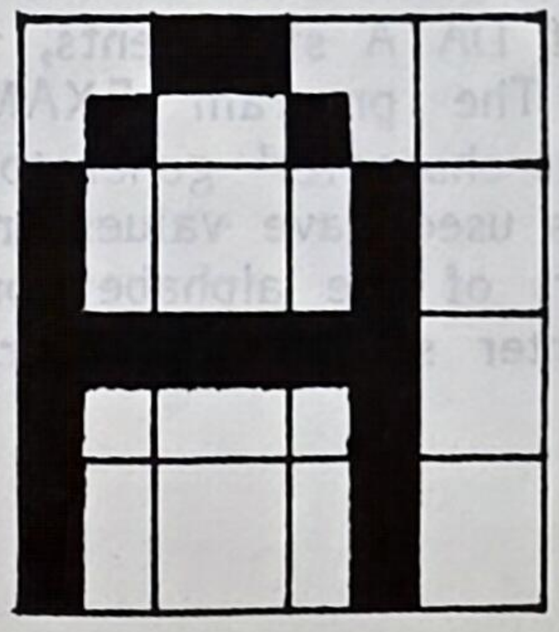


DISPLAYING AND MOVING 'MACRO CHARACTERS' 7

The POKE command can be used to display and move single characters and the PRINT command can be used to display and move lines of characters, but how does one display and move a block of characters a few characters wide and several lines deep. The ability to display such a block of characters is very valuable in many graphics applications. Pictorial displays can be constructed from a fairly small number of character blocks, some of which may be used more than once in the display. This is identical to the function of the character generator within the PET video circuitry which converts a simple code into a complex array of dots on the screen which comprise a character. A character generator reduces memory requirements from 8K to 1K for the video RAM on a 40 column machine. If all the blocks of characters are the same size, say 4 characters wide by 4 lines deep, we can call a set of blocks a 'macro-character set'. Such a character set can then be used in the same manner that the conventional character set is used, to display a large character (ie a display block) at any location on the screen. The advantage is that a whole block of characters can be displayed just by giving the code value for the block, memory saving becomes even greater when that same block is used more than once.

Though 'macro character sets' can be generated and displayed using subroutines written in Basic (this is done in the program EXAMPLE 19 which generates a moving signboard display) the process is much faster and more conveniently done in machine code. The program is stored at the top of memory starting at location 31285, and the data used by the program - the table of 'macro characters' - is stored from location 28673 up. This gives enough space to store 64 'macro characters' each 4 by 4 characters, character size is programmer definable and can be set to any reasonable values. To display a block all that is required are the screen coordinates of the top left corner of the block and the block number, these three variables are stored using POKE commands in specified memory locations.

The design of each 'macro character' requires very careful thought and planning, the methods used are the same as those used for planning the layout of the whole screen. The first stage is to lay each character out on squared paper thus:



Having designed the 'macro-character' each component character should be converted into its ASCII code value using tables at the back of this book, remember that it is the screen ASCII code which is required not the internal version of ASCII, the character again laid out on squared paper will look like this:

108	226	123	32
97	32	225	32
236	226	251	32
97	32	225	32

The reason for doing this is that the 'macro-character' must be converted into a memory map, for inclusion into the character generator look up table used by the program. The look up table is located in a 1K block of memory starting at address 28672, each 'macro-character' occupying 16 consecutive memory locations (for a 4x4 character size). The order of the characters occupying these locations is important and must be entered correctly if the 'macro-character' is to be displayed properly. The order used is that the lowest address contains the top right character then working from right to left and top to bottom until the highest address contains the bottom left character.

1	2	3	4
5	6		8
9	10	11	12
13	14	15	16

The macro-character look up table can be loaded into the correct memory locations using a simple loader from a Basic program. The ASCII codes for each of the component characters are stored as a set of DATA statements, these are simply read and poked into memory. The program EXAMPLE 20 shows the loading of the 'macro-character' generator look up table, the 'macro character' codes used have values from 0 to 26 and will display the letters of the alphabet on a 4 x 4 character block drawn using quarter square characters.

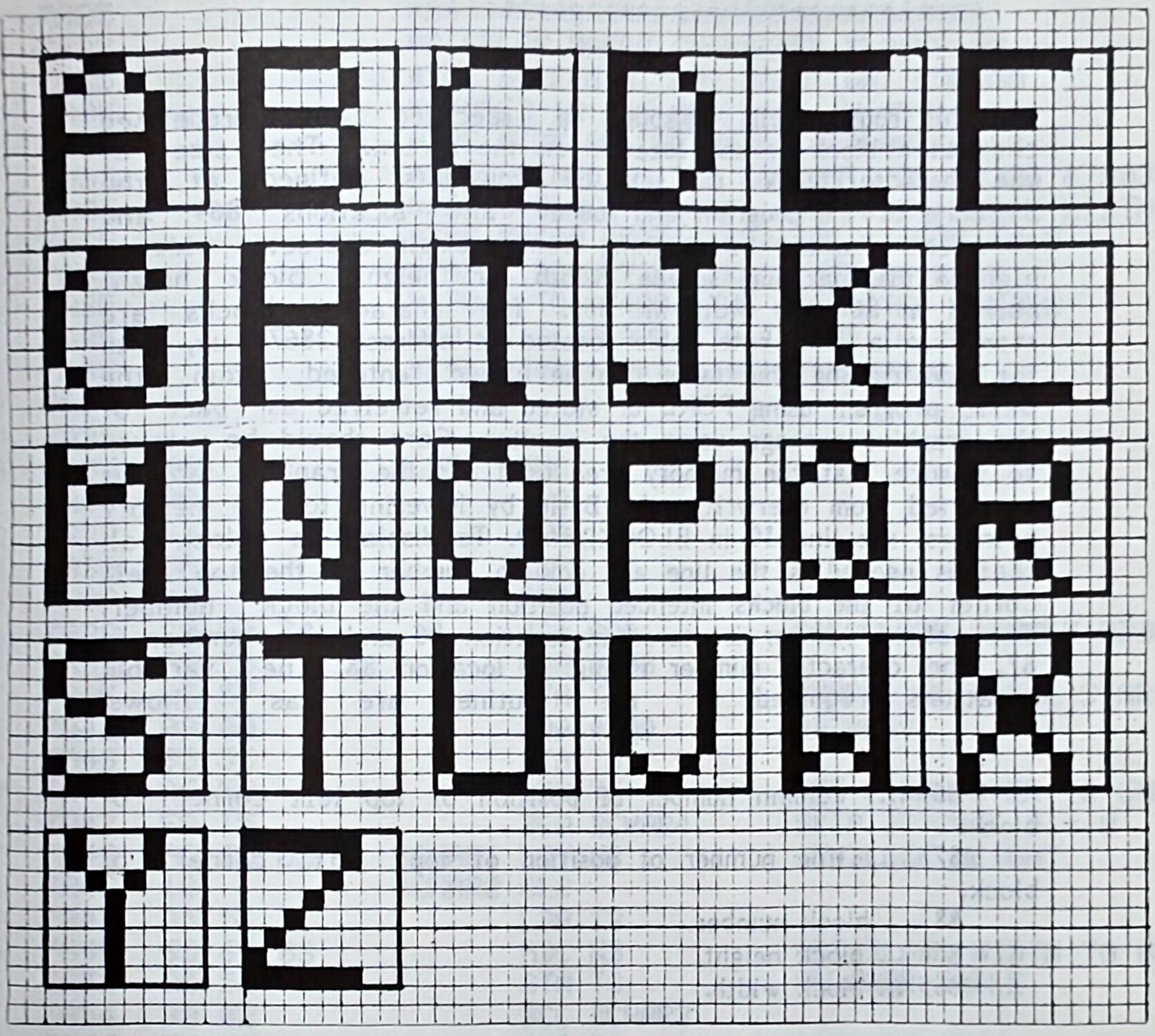


FIG 11. 'MACRO CHARACTER SET' USED IN EXAMPLE 11

Character Block Display.

This routine will display a specified and predefined character block at any location on the screen. The size of the character block is set by constants defined at the begining of the program and loaded into locations 864 and 865. The size can be anything from a single character wide or high to the maximum screen width or height. Block height stored in 864 and width in 865. The character blocks are stored in a 1000 byte table located at address 28673 up, data for constructing this table can be either entered from the Basic program using POKE or stored and retreived as part of the graphics package using the monitor. Care should be taken to ensure that the memory area used for the graphics table is protected from overwriting by Basic by lowering top of memory pointers, see line 10 in BLOCKDEMO. To display a block all that is needed is the line and column number of the top left corner of the blocks intended position and the block number. The column number is stored in location 86 and line number in 87, the character number is put in location 88. The variable locations required by the routine are as follows:

86 .... column number of position of top left corner of block.

87 .... line number of position of top left corner of block.

88 .... block number.

864 ... block height.

865 ... block width.

28673 to 29672 .... 1000 byte table for storage of character block data.

The routine is called from a Basic program with the command:

SYS(31285)

The following is an example of a Basic program using this routine:

```
10 POKE 864,4   : REM SET BLOCK HEIGHT TO 4 CHARACTERS
20 POKE 865,4   : REM SET BLOCK WIDTH TO 4 CHARACTERS
30 REM
40 REM
100 POKE 86,10 : REM COLUMN NUMBER OF TOP LEFT CORNER OF BLOCK = 10
110 POKE 87,5  : REM LINE NUMBER OF TOP LEFT CORNER OF BLOCK = 5
120 POKE 88,12 : REM DISPLAY BLOCK NUMBER 12 AT PREDEFINED POSITION
```

```
LINE# LOC    CODE          LINE

1286  7A35                ;**********************************
1287  7A35                ;*DISPLAY MACRO CHARACTER BLOCKS ON
1288  7A35                ;*THE SCREEN AT ANY LOCATION, COLUMN
1289  7A35                ;*STORED IN $56 AND LINE IN $57.
1290  7A35                ;*THE SIZE OF THE MACRO CHARACTER IS
1291  7A35                ;*DETERMINED BY THE CONTENTS OF $0350 & $0351
1292  7A35                ;*$0350=BLOCK HEIGHT, AND $0351=BLOCK WIDTH
1293  7A35                ;*THE MACRO CHARACTERS ARE STORED IN A
1294  7A35                ;*TABLE OCCUPYING 1K OF MEMORY FROM
1295  7A35                ;*$7000 TO $7400. THE CHARACTER NUMBER
1296  7A35                ;*TO BE DISPLAYED IS STORED IN $58
1297  7A35                ;**********************************
1298  7A35                ;
1299  7A35                ;
1300  7A35  48            BCHAR  PHA
1301  7A36  98                   TYA
1302  7A37  48                   PHA
1303  7A38  8A                   TXA
1304  7A39  48                   PHA
1305  7A3A  20 00 74             JSR ADR1        ;CALCULATE SCREEN ADDRESS
1306  7A3D  A9 70                LDA #$70
1307  7A3F  85 57                STA T4          ;CHARACTER TABLE START
1308  7A41  A9 00                LDA #$00        ;ADDRESS
1309  7A43  85 56                STA T3
1310  7A45  A6 58                LDX T5          ;SET X REGISTER TO CHAR #
1311  7A47  F0 15                BEQ BCHAR4      ;SKIP IF CHAR # = 0
1312  7A49  AC 60 03      BCHAR1 LDY HB           ;Y=BLOCK HEIGHT
1313  7A4C  18            BCHAR2 CLC
1314  7A4D  A5 56                LDA T3
1315  7A4F  6D 61 03             ADC WB          ;ADD BLOCK WIDTH TO ADDRESS
1316  7A52  85 56                STA T3          ;OF CHAR TABLE
1317  7A54  90 02                BCC BCHAR3
1318  7A56  E6 57                INC T4
1319  7A58  88            BCHAR3 DEY
1320  7A59  D0 F1                BNE BCHAR2
1321  7A5B  CA                   DEX             ;ANOTHER BLOCK
1322  7A5C  D0 EB                BNE BCHAR1
1323  7A5E  AE 60 03      BCHAR4 LDX HB          ;X=BLOCK HEIGHT
1324  7A61  AC 61 03      BCHAR5 LDY WB          ;Y=BLOCK WIDTH
1325  7A64  B1 56         BCHAR6 LDA (T3),Y
1326  7A66  91 54                STA (T1),Y      ;DISPLAY BYTE
1327  7A68  88                   DEY
1328  7A69  D0 F9                BNE BCHAR6      ;NEXT
1329  7A6B  18                   CLC
1330  7A6C  A5 56                LDA T3
1331  7A6E  6D 61 03             ADC WB          ;ADD WIDTH OFFSET
1332  7A71  85 56                STA T3
1333  7A73  90 02                BCC BCHAR7
1334  7A75  E6 57                INC T4
1335  7A77  18            BCHAR7 CLC
1336  7A78  A5 54                LDA T1
1337  7A7A  69 28                ADC #$28        ;ADD SCREEN WIDTH OFFSET
1338  7A7C  85 54                STA T1
1339  7A7E  90 02                BCC BCHAR8
1340  7A80  E6 55                INC T2
```

```
LINE# LOC    CODE          LINE

1341  7A82   CA          BCHAR8 DEX                  ;NEXT LINE OF BLOCK
1342  7A83   D0 DC              BNE BCHAR5
1343  7A85   68                 PLA
1344  7A86   AA                 TAX
1345  7A87   68                 PLA
1346  7A88   A8                 TAY
1347  7A89   68                 PLA
1348  7A8A   60                 RTS
1349  7A8B               ;
1350  7A8B               ;
```

EXAMPLE 19

```
1 REM ************************************
2 REM *MOVING SIGNBOARD - WILL DISPLAY A STRING INPUT AT THE
3 REM *START OF THE PROGRAM IN LARGE CHARACTERS MOVING ACROSS
4 REM *THE SCREEN FROM RIGHT TO LEFT, THIS LEFT SCROLLING IS
5 REM *DONE BY A MACHINE CODE ROUTINE STORED IN CASSETTE BUFFER
6 REM *#2 BY THE PROGRAM - FIRST 16 LINES OF CODE.
9 REM ************************************
10 DATA826,50
20 DATA162,0,189,65,129,157,64,129
30 DATA189,105,129,157,104,129
40 DATA189,145,129,157,144,129
50 DATA189,185,129,157,184,129
60 DATA189,225,129,157,224,129
70 DATA189,9,130,157,8,130
80 DATA189,49,130,157,48,130
90 DATA232,224,39,144,209,96
100 REM LOAD TAPE BUFFER #2 WITH SCAN CODE
110 READ AD,NO
120 FORNN=1TO NO
130 READ NX
140 POKE AD,NX
150 AD=AD+1
160 NEXT NN
170 PRINT"□"
180 PRINT"INPUT STRING"
190 INPUT IN$
200 GOSUB1160
210 GOSUB490
220 GOSUB240
230 GOTO220
240 REM DISPLAY STRING ON SCREEN
250 FOR X=1TOLEN(IN$)
260 X1=ASC(MID$(IN$,X,1))-32
270 IFX1<0ORX1>63THEN380
280 GOSUB420
290 FORY=1TO29STEP7
300 GOSUB 400
310 XD=33367
320 FORZ=0TO6
330 X4=ASC(MID$(Z$(X1),Y+Z,1)):POKEXD,X4
340 XD=XD-40
350 NEXTZ
360 NEXTY
370 GOSUB400
380 NEXTX
390 RETURN
400 SYS(826)
410 RETURN
420 REM SPACE BETWEEN LETTERS
430 XE=33367
440 FORV=0TO6
450 POKEXE,ASC(" ")
460 XE=XE-40
```

```
470 NEXTV
480 RETURN
490 REM SET-UP LARGE CHARACTER ARRAY
500 DIMZ$(63)
510 Z$(00)="                                                "
520 Z$(01)="         Q QQQQQ                                "
530 Z$(02)="            QQQ              QQQ                "
540 Z$(03)="  Q Q   QQQQQQQ  Q Q  QQQQQQQ  Q Q  "
550 Z$(04)=" Q  QQ Q  Q  QQQQQQQQQ  Q  Q QQ  Q "
560 Z$(05)="QQ   QQ  Q  QQ   Q   QQ  Q  QQ   QQ"
570 Z$(06)=" Q  QQ Q QQ  QQ QQ  Q Q  QQ Q Q    "
580 Z$(07)="            QQQ                                 "
590 Z$(08)="          QQQ   Q   Q Q     Q                  "
600 Z$(09)="         Q      Q Q   Q   QQQ                   "
610 Z$(10)=" Q Q Q   QQQ  QQQQQQQ  QQQ   Q Q Q "
620 Z$(11)="  Q       Q    QQQQQ    Q      Q   "
630 Z$(12)="        Q Q     QQ                              "
640 Z$(13)="  Q       Q       Q       Q      Q   "
650 Z$(14)="        QQ      QQ                              "
660 Z$(15)="QQ          Q       Q       Q        QQ"
670 Z$(16)="QQQQQQQQ      QQ      QQ      QQQQQQQQ"
680 Z$(17)="          Q    Q QQQQQQQQ                       "
690 Z$(18)="QQQQ Q Q  Q  QQ  Q  QQ  Q  QQ   QQ "
700 Z$(19)=" Q   Q Q  Q  QQ  Q  QQ  Q  Q QQ QQ "
710 Z$(20)="   QQQ    Q  QQQQQQQQ   Q       Q   "
720 Z$(21)=" Q QQQQQ  Q  QQ  Q  QQ  Q  Q QQ   Q"
730 Z$(22)=" QQQQQ Q  Q  QQ  Q  QQ  Q  Q QQ  Q "
740 Z$(23)="QQ    Q  Q  Q   Q   Q    Q Q      QQ"
750 Z$(24)=" QQ QQ Q  Q  QQ  Q  QQ  Q  Q QQ QQ "
760 Z$(25)=" Q  QQ Q  Q  QQ  Q  QQ  Q  Q QQQQQ "
770 Z$(26)="          QQ QQ  QQ QQ                          "
780 Z$(27)="         Q Q QQ  QQ QQ                          "
790 Z$(28)="    Q      Q Q   Q   Q Q      Q                "
800 Z$(29)="  Q Q     Q Q    Q Q    Q Q    Q Q  "
810 Z$(30)="Q      Q Q   Q   Q Q      Q                    "
820 Z$(31)="     Q        QQ Q   Q   Q  Q    QQ "
830 Z$(32)=" QQQQQ Q      QQ QQQ QQ Q Q QQ QQQQQ"
840 Z$(33)="QQQQQ      Q Q    Q  Q   Q Q QQQQQ  "
850 Z$(34)="QQQQQQQQ  Q  QQ  Q  QQ  Q  Q QQ QQ "
860 Z$(35)=" QQQQQ Q      QQ      QQ      Q Q  Q "
870 Z$(36)="QQQQQQQQ      QQ      QQ      Q QQQQQ "
880 Z$(37)="QQQQQQQQ  Q  QQ  Q  QQ  Q  QQ      Q"
890 Z$(38)="QQQQQQQ   Q  Q   Q  Q   Q  Q      Q"
900 Z$(39)=" QQQQQ Q      QQ      QQ  Q  Q QQQ Q "
910 Z$(40)="QQQQQQQ   Q        Q       Q   QQQQQQQ"
920 Z$(41)="         Q      QQQQQQQQQ      Q          "
930 Z$(42)=" Q    QQ      Q QQQQQQ      Q       Q"
940 Z$(43)="QQQQQQQ   Q      Q Q   Q   Q Q      Q"
950 Z$(44)="QQQQQQQQ      Q       Q       Q       "
960 Z$(45)="QQQQQQQ     Q    QQ        Q QQQQQQQ"
970 Z$(46)="QQQQQQQ     Q    QQ    Q    QQQQQQQ"
980 Z$(47)=" QQQQQ Q      QQ      QQ      Q QQQQQ "
990 Z$(48)="QQQQQQQ   Q  Q   Q  Q   Q  Q    QQ "
1000 Z$(49)=" QQQQQ Q      QQ Q   QQQ    QQQQQQQ "
1010 Z$(50)="QQQQQQQ   Q  Q  QQ  Q Q Q  QQ   QQ "
1020 Z$(51)=" Q  QQ Q  Q  QQ  Q  QQ  Q  Q QQ  Q "
1030 Z$(52)="       Q       QQQQQQQQ       Q       Q"
1040 Z$(53)=" QQQQQQQ       Q       Q       QQQQQQ"
1050 Z$(54)="   QQQQ QQ     Q         QQ        QQQQ"
1060 Z$(55)="QQQQQQQ Q        QQ     Q      QQQQQQQ"
```

```
1070 Z$(56)="QQ    QQ  Q Q     Q      Q Q  QQ   QQ"
1080 Z$(57)="      QQ   QQ  QQQ        QQ       QQ"
1090 Z$(58)="QQ     QQ Q   QQ  Q  QQ    Q QQ     QQ"
1100 Z$(59)="        QQQQQQQQ     QQ      Q       "
1110 Z$(60)="      QQ    Q      Q      Q    QQ     "
1120 Z$(61)="        Q     QQ      QQQQQQQQ        "
1130 Z$(62)="    Q          Q QQQQQQQ      Q     Q  "
1140 Z$(63)="QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"
1150 RETURN
1160 REM MAKE SCREEN GREY
1170 PRINT"■"
1180 FOR M=1TO1000:PRINT"▒";:NEXT
1190 RETURN
READY.
```

EXAMPLE 20

```
10 REM ************************************
20 REM *DEMONSTRATION PROGRAM TO SHOW USE
30 REM *OF MACRO CHARACTER BLOCK ROUTINE
40 REM ************************************
50 REM PROTECT TOP OF MEMORY
60 POKE52,00:POKE53,112
70 POKE48,00:POKE49,112:CLR
80 REM DEFINE HEIGHT AND WIDTH OF MACRO BLOCKS
90 POKE864,4:POKE865,4
100 PRINT"□"
110 REM
120 REM LOAD TABLE OF MACRO CHARACTER DATA
130 REM INTO MEMORY
140 REM
150 REM 'SPACE' CHARACTER
160 DATA 32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
170 REM 'A'
180 DATA 108,226,123,32,97,32,225,32,236,226,251,32,97,32,225,32
190 REM 'B'
200 DATA 236,226,127,32,252,98,255,32,97,32,225,32,252,98,255,32
210 REM 'C'
220 DATA 108,226,127,32,97,32,32,32,97,32,32,32,124,98,255,32
230 REM 'D'
240 DATA 236,226,123,32,97,32,225,32,97,32,225,32,252,98,126,32
250 REM 'E'
260 DATA 236,226,226,32,252,98,32,32,97,32,32,32,252,98,98,32
270 REM 'F'
280 DATA 236,226,226,32,252,98,32,32,97,32,32,32,97,32,32,32
290 REM 'G'
300 DATA 108,226,127,32,97,32,32,32,97,32,251,32,124,98,255,32
310 REM 'H'
320 DATA 97,32,225,32,252,98,254,32,97,32,225,32,97,32,225,32
330 REM 'I'
340 DATA 32,251,126,32,32,225,32,32,32,225,32,32,32,254,123,32
350 REM 'J'
360 DATA 32,124,236,32,32,32,97,32,32,32,97,32,127,98,126,32
370 REM 'K'
380 DATA 97,32,255,32,252,255,32,32,236,127,32,32,97,32,127,32
390 REM 'L'
400 DATA 97,32,32,32,97,32,32,32,97,32,32,32,252,98,98,32
```

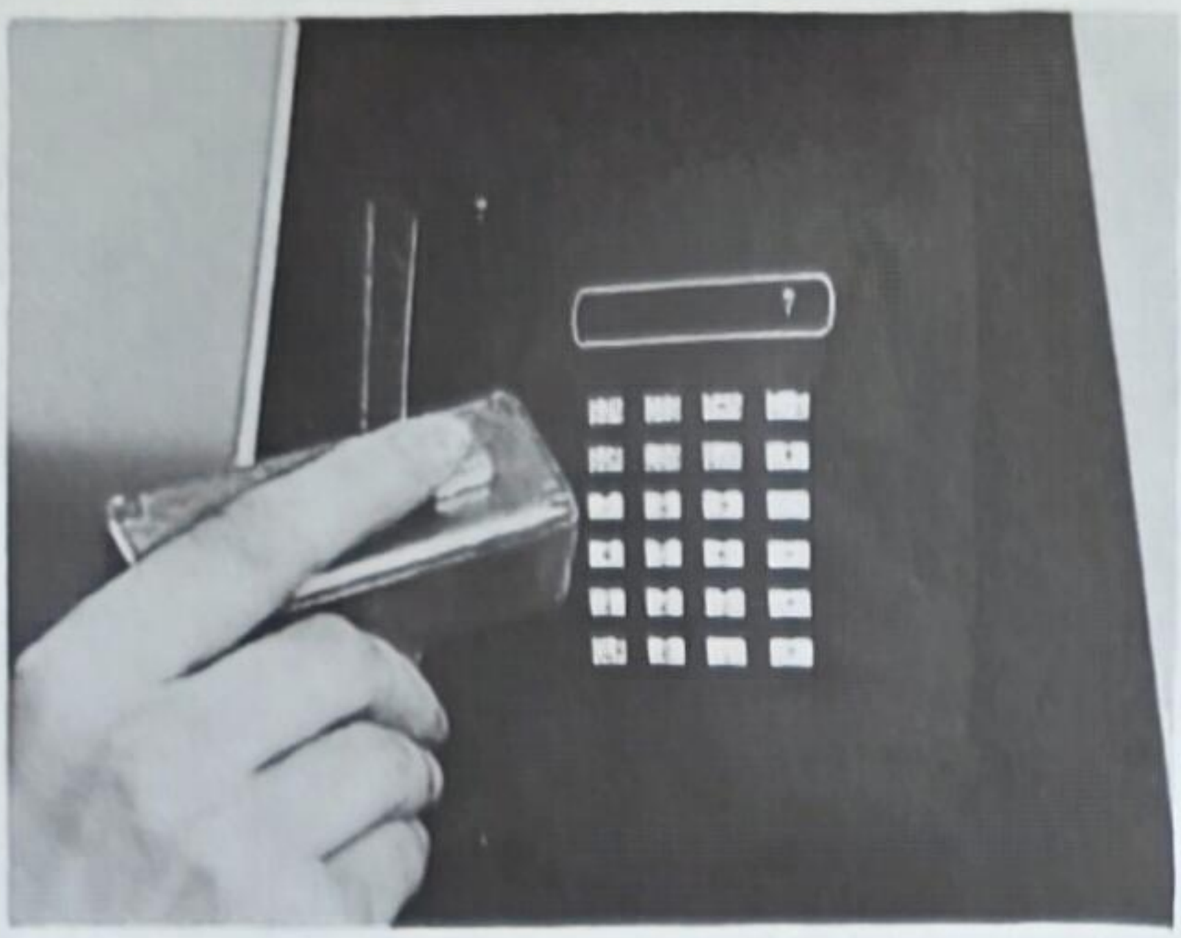
```
410 REM 'M'
420 DATA 252,32,254,32,97,226,225,32,97,32,225,32,97,32,225,32
430 REM 'N'
440 DATA 252,32,225,32,97,97,225,32,97,225,225,32,97,32,251,32
450 REM 'O'
460 DATA 108,226,123,32,97,32,225,32,97,32,225,32,124,98,126,32
470 REM 'P'
480 DATA 236,226,127,32,252,98,255,32,97,32,32,32,97,32,32,32
490 REM 'Q'
500 DATA 108,226,123,32,97,32,225,32,97,108,225,32,124,98,127,32
510 REM 'R'
520 DATA 236,226,127,32,252,98,255,32,97,127,32,32,97,32,127,32
530 REM 'S'
540 DATA 255,226,127,32,127,123,32,32,32,124,127,32,127,98,255,32
550 REM 'T'
560 DATA 124,251,226,32,32,225,32,32,32,225,32,32,32,225,32,32
570 REM 'U'
580 DATA 97,32,225,32,97,32,225,32,97,32,225,32,127,98,255,32
590 REM 'V'
600 DATA 97,32,225,32,97,32,225,32,97,32,225,32,124,98,126,32
610 REM 'W'
620 DATA 97,32,225,32,97,32,225,32,97,98,225,32,236,32,251,32
630 REM 'X'
640 DATA 97,32,225,32,124,98,126,32,108,226,123,32,97,32,225,32
650 REM 'Y'
660 DATA 225,32,225,32,32,127,126,32,32,225,32,32,32,225,32,32
670 REM 'Z'
680 DATA 226,226,251,32,32,108,126,32,108,126,32,32,252,98,98,32
690 DATA 1000
700 Q=28673
710 READ A
720 IF A=1000THEN760
730 POKEQ,A
740 Q=Q+1
750 GOTO710
760 REM
770 REM DISPLAY THE WORDS 'BIG PET TEXT' IN LARGE CHARACTERS
780 REM THEY MOVE FROM TOP RIGHT TO THEIR PROPER LOCATION
790 REM
800 REM
810 DATA 16,16,4
820 DATA 5,16,8
830 DATA 20,16,12
840 DATA 20,16,20
850 DATA 5,16,24
860 DATA 24,16,28
870 DATA 20,16,32
880 DATA 2,8,14
890 DATA 9,8,18
900 DATA 7,8,22
910 FORX=1TO10
920 READ C,H,W
930 FORW1=1TOW
940 POKE86,W1-1:POKE87,1:POKE88,0
950 SYS(31285)
960 POKE86,W1:POKE87,1:POKE88,C
970 SYS(31285)
980 FORQ=1TO50:NEXTQ
990 NEXTW1
1000 FORH1=1TOH
```

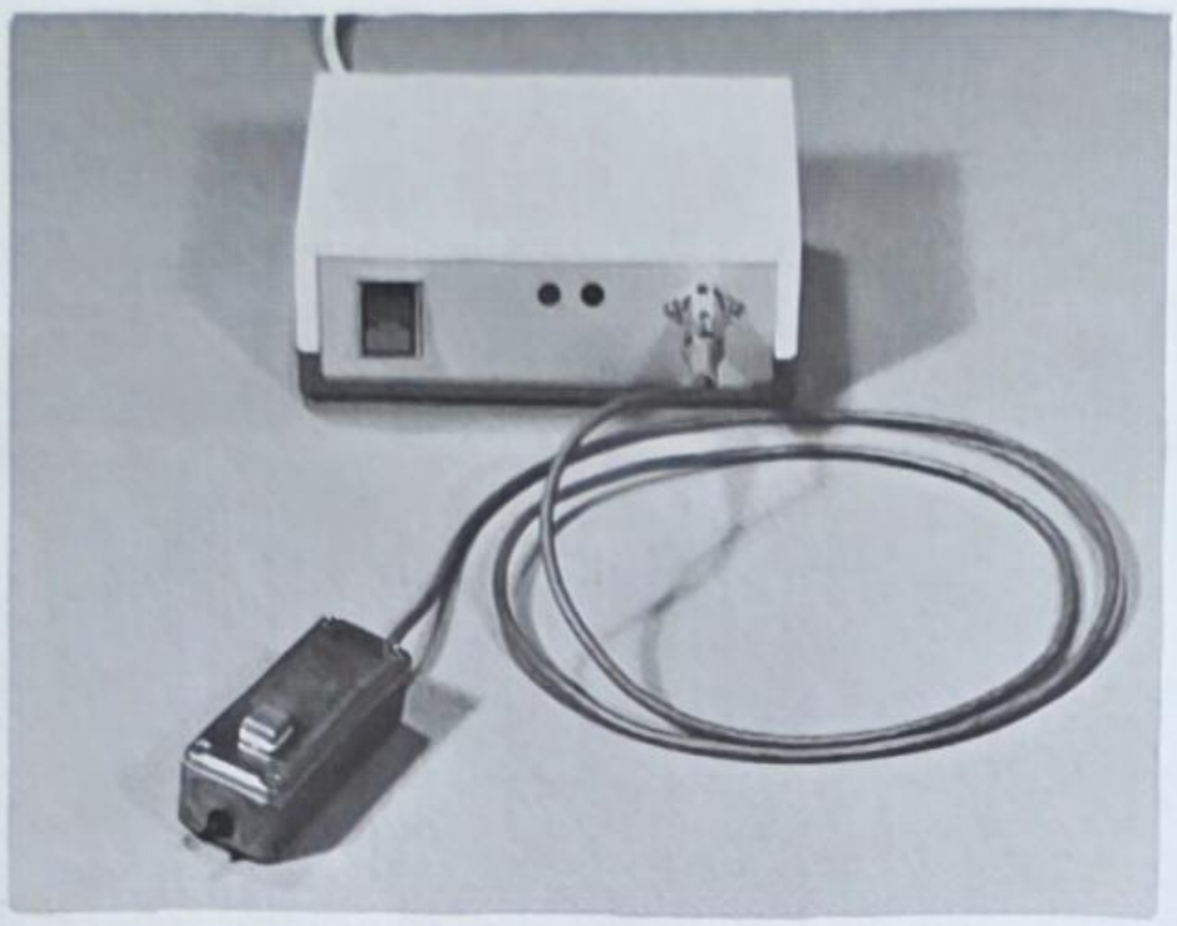
```
1010 POKE86,W:POKE87,H1-1:POKE88,0
1020 SYS(31285)
1030 POKE86,W:POKE87,H1:POKE88,C
1040 SYS(31285)
1050 FORQ=1TO50:NEXTQ
1060 NEXTH1
1070 NEXTX
1080 FORQ=1TO5000:NEXTQ
1090 REM
1100 REM ROUTINE TO ALLOW THE USER TO PUT MACRO CHARACTERS
1110 REM ANYWHERE ON THE SCREEN USING X AND Y
1120 REM COORDINATES AND CHARACTER NUMBER
1130 REM
1140 PRINT"□";
1150 INPUT"X COORDINATE ";X
1160 INPUT"Y COORDINATE ";Y
1170 INPUT"CHARACTER # ";C
1180 IFX=255THENEND
1190 POKE 86,Y:POKE87,X:POKE88,C
1200 SYS(31285)
1210 PRINT"■";:GOTO1150
READY.
```

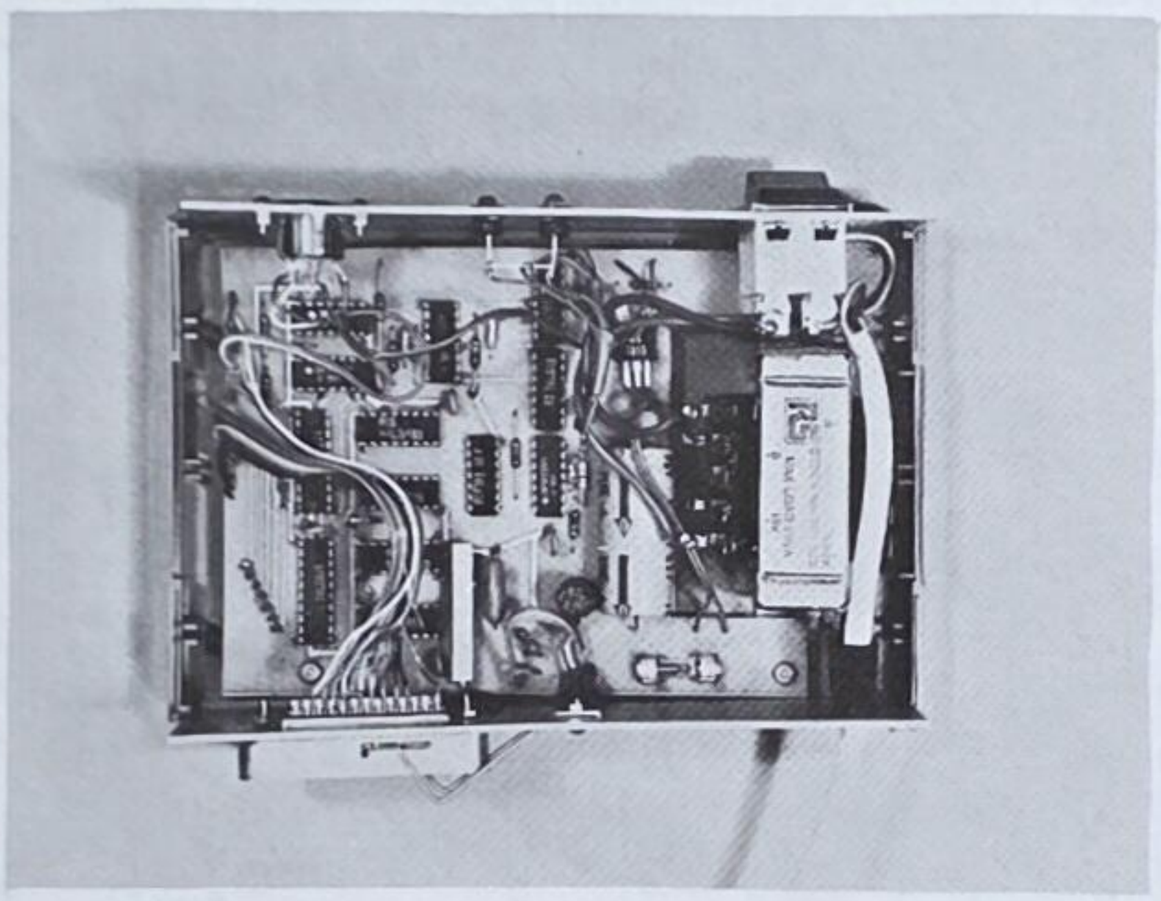
Notes

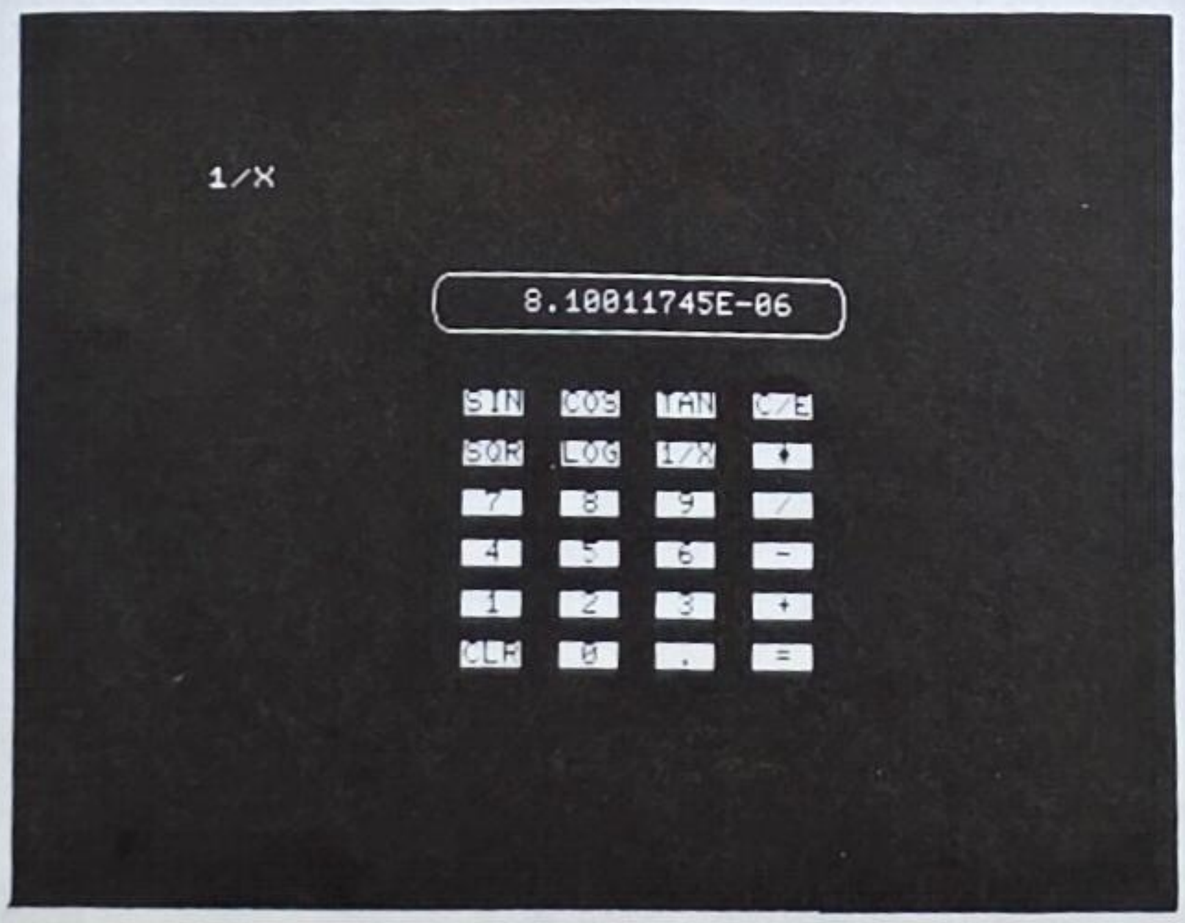
8 A LIGHT PEN FOR THE PET

Principles of construction of a light pen for the PET - 177
Construction of a light pen - 182
Programming the PET to use a light pen - 182
Notes on setting up and using the light pen - 188
A machine code package for the light pen - 191
Example program to show principles of light pen operation - 195
Program to test the operation of the light pen - 195
Program to demonstrate use of light pen and machine code - 196
Example use of light pen, emulates pocket calculator - 201








1/X
8.10011745E-06
SIN
COS
TAN
C/E
SQR
LOG
1/X
7
8
9
4
5
6
1
2
3
CLR
0
.
=

A LIGHT PEN FOR THE PET

8

A light pen is a device which allows the programmer to determine where on the video screen a user is pointing a special 'pen' unit, and direct his program accordingly. From the users point of view just touching the screen with the pen enters the required information into the computer. This information, in the form of X,Y coordinates, can be interpreted by the program and the appropriate action initiated. The program EXAMPLE 21 shows how such a device might operate. This is best demonstrated by connecting a normaly closed switch to line 0 of the user port, this switch simulates a photocell. Loop I and loop J POKE reverse spaces into all screen locations and then check to see if the 'photocell' has gone high, simulated by the switch being pressed. If not, then the character position reverts to a normal space. If after all possible screen locations have been tested and line 0 is still low then a message to that effect is printed. If the switch (photocell) was pressed then the values for I and J are printed, these correspond to the line and column number being tested when the switch was pressed, (when the cursor reached the spot pointed to by the pen). However, this method is totaly inadiquate, since it takes over 40 seconds to traverse the whole screen and a photocell by itself is unable to distinguish between the weak light from the screen and ambient room lighting.

Fortunatly the information on the screen is alrea y scanned. The PET display is created by an electron beam scanned rapidly across and down the screen, this is explained in the first section of this book. It takes less than 70 microseconds for each raster line to traverse the screen and data actualy appears on the screen for only 40 microseconds of this time, or just one microseconds to cover the width of one character. The whole screen is displayed in about 17 milliseconds and this is repeated every 1/60 of a second. These timings are far shorter than those which can be handled by a microprocessor under software control, even in machine code. Since software control of a lightpen is impractical it is necessary to implement the system using high speed electronics.

All the signals required by such a hardware system are available from the user port on the PET. These indicate the start of each line and the begining of each frame, they are the line, field and sync pulses. As the timing for all the video signals in the PET are derived from the 8 MHz clock, this clock can be used to count along a raster line and record the position of the pen on that line. On the user port the highest clock frequency available is 125KHz, from the internal counter/timer feeding the CB2 line. As a quarter graphics point lasts only 0.5 microseconds the frequency on CB2 must be multiplied by 16. This is acheved by using a

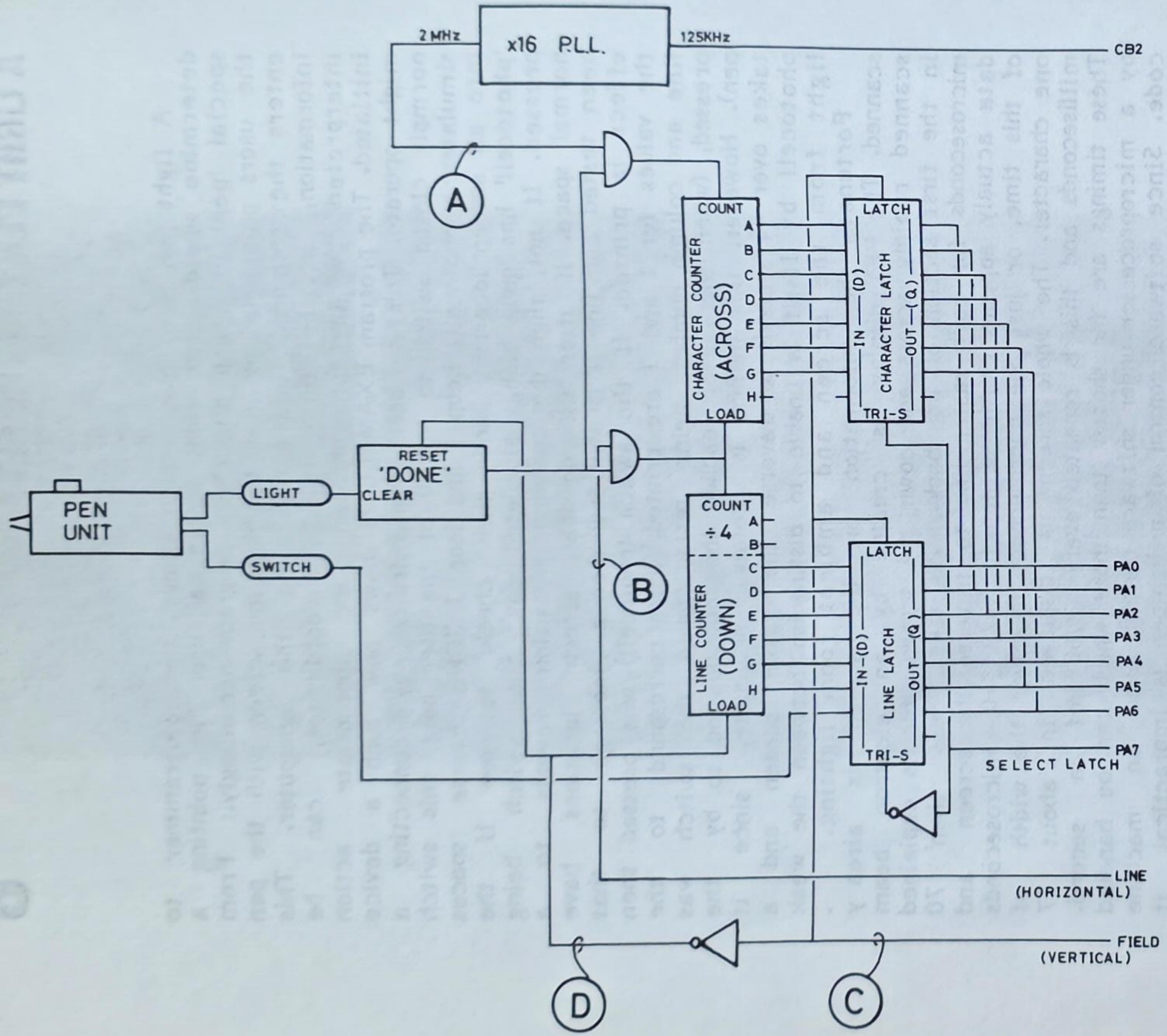

2 MHz
x16 P.L.L.
125KHz
CB2
A
COUNT
CHARACTER COUNTER
(ACROSS)
A
B
C
D
E
F
G
H
LOAD
LATCH
IN (D)
CHARACTER LATCH
OUT (Q)
TRI-S
RESET
'DONE'
CLEAR
LIGHT
PEN
UNIT
SWITCH
COUNT
÷4
A
B
LINE COUNTER
(DOWN)
C
D
E
F
G
H
LOAD
B
LATCH
IN-(D)
LINE LATCH
OUT-(Q)
TRI-S
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
SELECT LATCH
LINE
(HORIZONTAL)
FIELD
(VERTICAL)
D
C

FIG 12.

phase locked loop frequency multiplier. The vertical position of the pen is determined by counting the number of lines from the start of the frame to the point where the pen detects light from the screen.

A block diagram of the light pen unit is shown in Fig 12 the relevant signals available at the user port and their timings are shown in Fig 13 The cycle starts at point 'D' where the line counter is cleared and the 'Done' latch is set to one. While this latch remains high 'A', pulses from the phase locked loop will increment the character counter once for every quarter graphics character position. At the begining of each line 'B' the character counter is cleared (actualy it is preloaded with 208 so that it reads zero just before the video information starts). Every fourth line, or once per quarter graphics character, the line counter is incremented.

If the light pen picks up light from the screen the 'Done' latch is cleared. This then inhibits the clock signals to the two counters and the load line of the character clock. Values in these counters are thus frozen to the value they contained when the light first appeared. Data in the counter is transfered to the output latches at point 'C' just before the cycle recomences at 'D'. This data remains stable for the duration of the next frame time, until 'C+1' and may be read by a program in the PET. If the latch 'Done' is not cleared by the pen, then the data values transfered to the latch are meaningless, this may be infered from the impossibly high line count. At the point 'C' the state of the 'Done' latch is stored in a second latch which illuminates an LED if valid data appears on the output latches.

One problem is how to transfer 13 bits of data (7 character and 6 line) to the PET via an 8 bit wide user port. The solution is to use latches with tri-state outputs. When the tri-state control input is taken high, the outputs go into a high impedance (apparently open circuit) state. When the tri-state control is taken low, data from the internal latch outputs appear on the physical output lines of the IC package. Many such outputs may be wired together so long as only one of them is enabled at once. This is the principle on which the data bus of a microprocessor is based. As only two outputs compete for a single input, a single inverter is sufficient to ensure that only one tri-state control is high while the other is low. Line 7 of the user port is configured as an output and used to control the tri-state outputs. Zero on PA7 selects the line counter, one selects the character counter. A switch is provided on the light pen unit itself, this is connected to the spare PA6 line on the line counter latch. This switch has no direct effect on the light pen hardware and is purely for the programmers convenience.

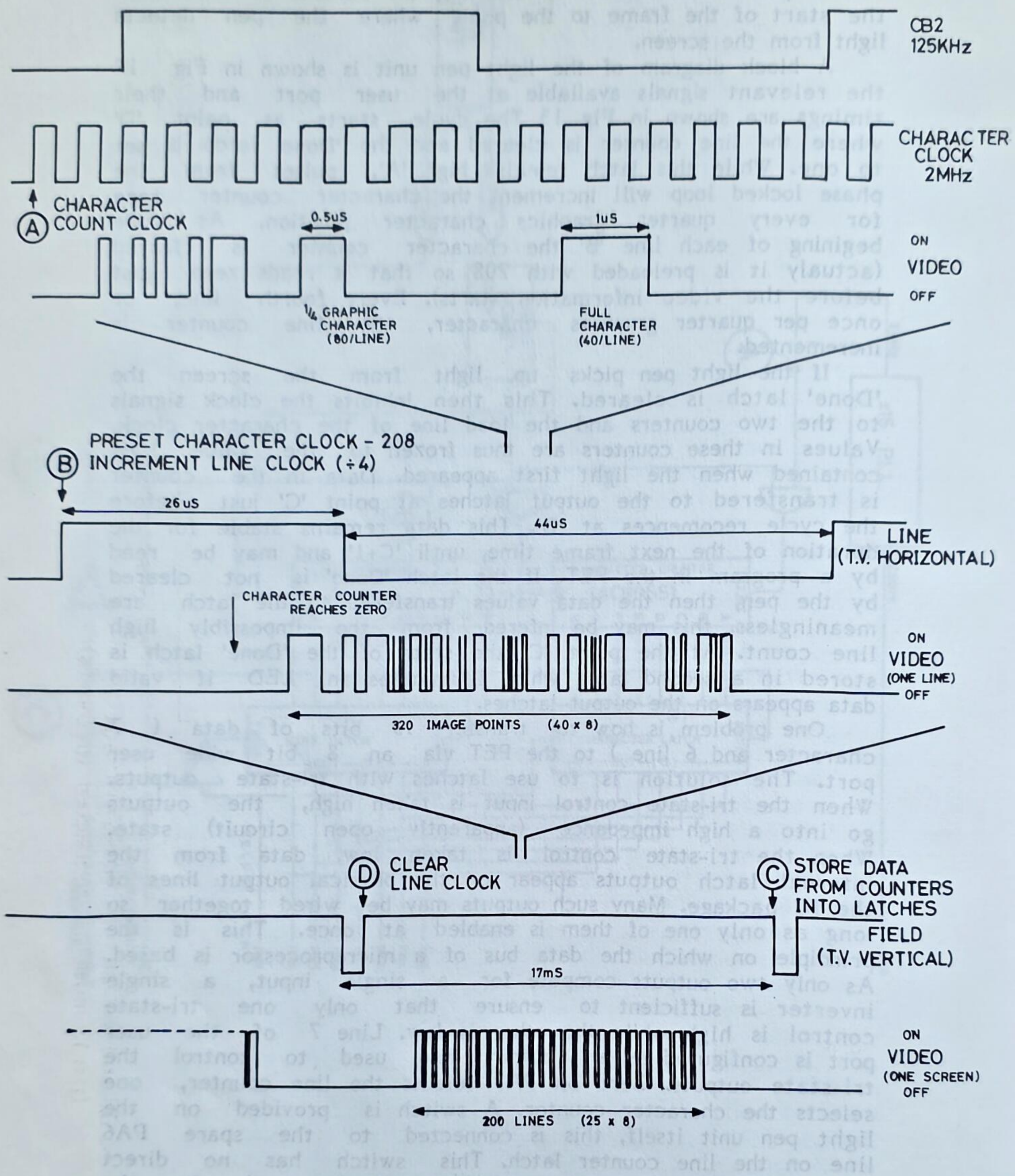
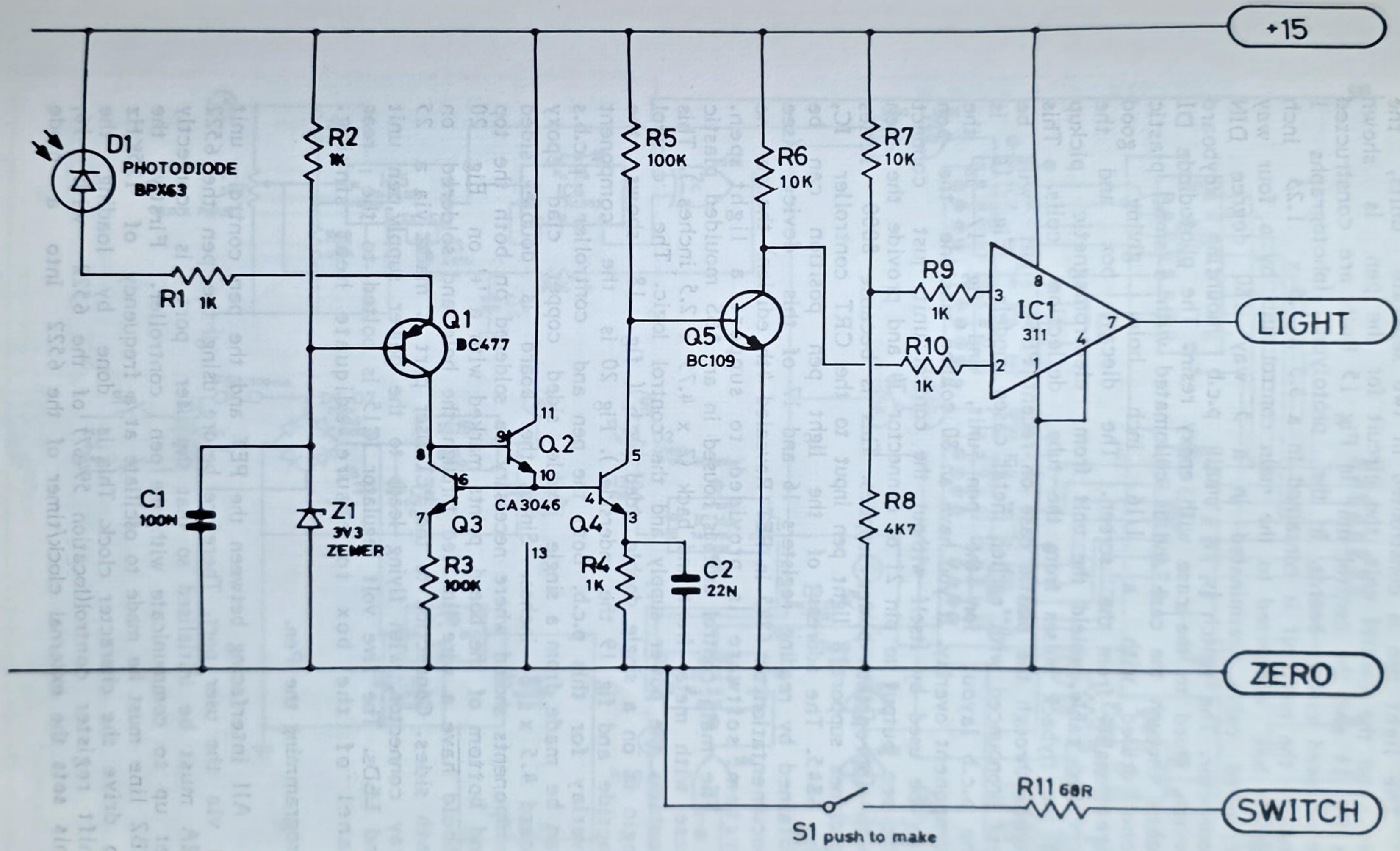

CB2
125KHz
CHARACTER
CLOCK
2MHz
A
CHARACTER
COUNT CLOCK
0.5uS
1uS
ON
VIDEO
OFF
¼ GRAPHIC
CHARACTER
(80/LINE)
FULL
CHARACTER
(40/LINE)
B
PRESET CHARACTER CLOCK - 208
INCREMENT LINE CLOCK (÷4)
26 uS
44uS
LINE
(T.V. HORIZONTAL)
CHARACTER COUNTER
REACHES ZERO
ON
VIDEO
(ONE LINE)
OFF
320 IMAGE POINTS (40 x 8)
D
CLEAR
LINE CLOCK
C
STORE DATA
FROM COUNTERS
INTO LATCHES
FIELD
(T.V. VERTICAL)
17mS
ON
VIDEO
(ONE SCREEN)
OFF
200 LINES (25 x 8)

FIG 13.



Q2, Q3 and Q4 all in CA3046 package
(pin 13 must be connected to 0volts)

FIG 14.

Construction.

The light pen is constructed in two seperate units, the pen, and the control unit, the circuit for the pen is shown in Fig 14 and the control unit in Fig 15 Both are constructed on printed circuit boards. In the prototype (photographs 1 and 2) the pen unit is contained in a 3.5 x 1.5 x 1.25 inch diecast box connected to the main control unit by a four way screened cable, terminated in a 5 way 180 degree DIN connector. The switch is a small p.c.b mounting keyboard switch, glued to the case with epoxy resin. The photodiode D1 pokes through the case and is collomated with a small plastic cone drilled with a 1/16 inch hole, giving good directionality from the screen. The diecast box and the screened cable, shield the unit from electromagnetic pickup of the flyback pulses from the tube deflection coils. This comes through the plastic case on recent PETs, but will be less pronounced with earlier metal cased models. Fig 16 is the p.c.b layout for the pen unit, and fig 17 is the component overlay. If you have an 80 column machine the pen can be used by itself without the control unit, just connect the pen output to pin 21 of connector J4 and provide the pen with appropriate power supplies. This is because 8000 series machines support a light pen input to the CRT controller IC, the 6845. The address of the light pen position can be obtained by reading registers 16 and 17 of this device (see documentation on 6845 in PET Revealed 4th edition) though no system software is provided to support a light pen.

The main control unit is housed in an ABS moulded plastic case with metal front and back (7 x 4.75 x 2.5 inches). This contains the power supply and the control logic. The control logic is on a single double sided p.c.b (fig 18 shows the topside and fig 19 the underside). Fig 20 is the component overlay for this p.c.b. Both the pen and controller p.c.b.s can be made from a single double sided copper clad epoxy board 4.5 x 8 inches. Since the board is double sided components should where necessary be soldered on both the top and bottom of the board, points marked with a '+' on Fig 20 should have a wire link fed through the hole and soldered on both sides. Connection to the PET user port is made via a 25 way connector, with flying leads to the power supply,pen unit and LEDs. The five volt regulator IC15 is bolted to the rear panel of the box to ensure adiquate heat sinking.

Programming the Pen.

All interfacing between the PET and the pen control unit is via the user port. Therefore before using the pen the 6522 VIA must be initialised so that the user port is correctly set up to communicate with the pen controller. Firstly the CB2 line must be made to oscillate at a frequency of 125KHz to drive the character clock. This is done by loading the shift register control(location 59467) of the 6522 with 16, this sets the external clock/timer of the 6522 into a mode

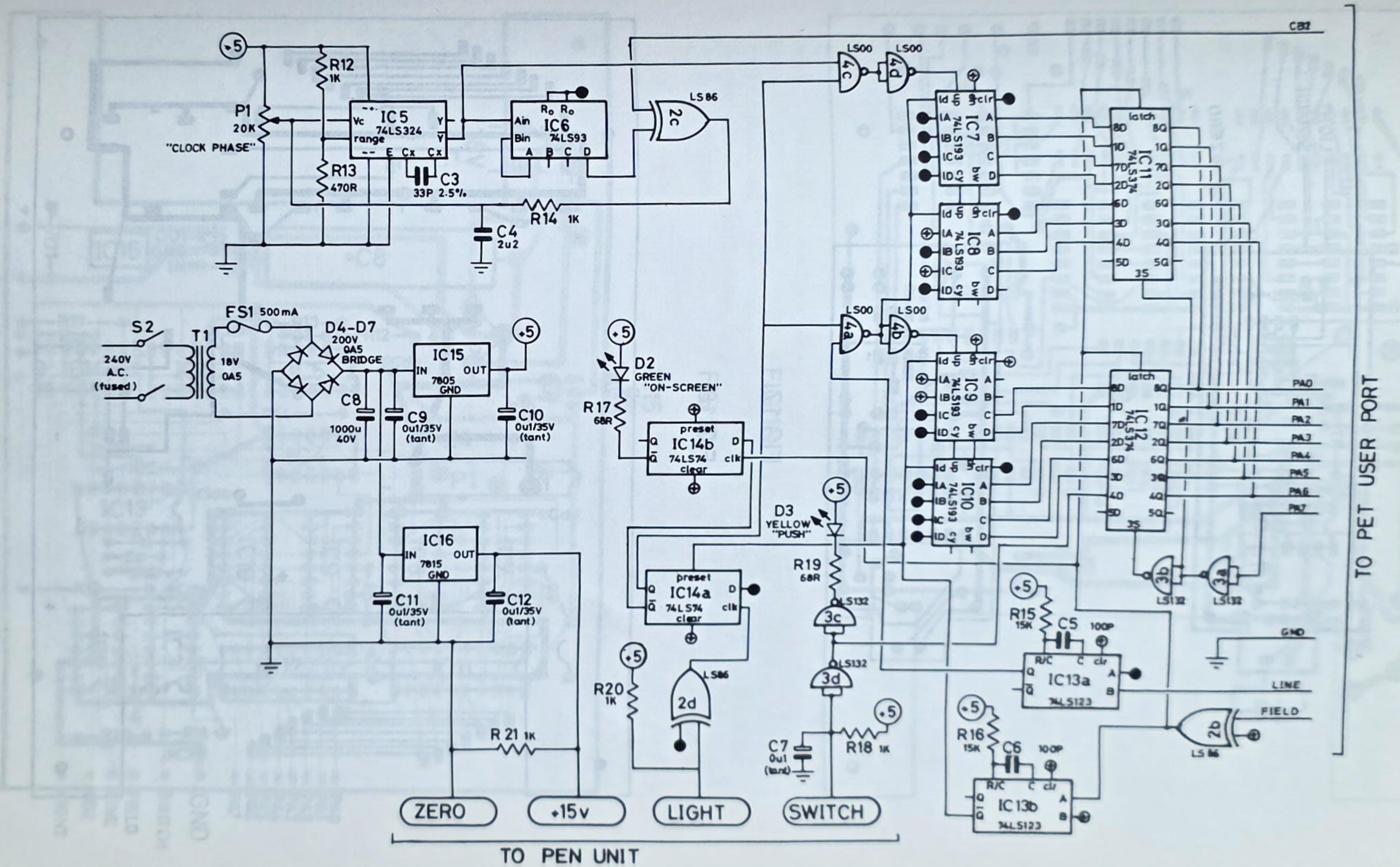

+5
R12 1K
P1 20K "CLOCK PHASE"
R13 470R
IC5 74LS324 range
C3 33P 2.5%
IC6 74LS93
LS86
R14 1K
C4 2u2
CB2
LS00
IC7 74LS193
IC8 74LS193
IC11 74LS374 latch
IC9 74LS193
IC10 74LS193
IC12 74LS374 latch
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
TO PET USER PORT
S2
240V A.C. (fused)
T1
FS1 500mA
18V 0A5
D4-D7 200V 0A5 BRIDGE
C8 1000u 40V
IC15 7805 GND
C9 0u1/35V (tant)
C10 0u1/35V (tant)
IC16 7815 GND
C11 0u1/35V (tant)
C12 0u1/35V (tant)
D2 GREEN "ON-SCREEN"
R17 68R
IC14b 74LS74 clear
D3 YELLOW "PUSH"
R19 68R
IC14a 74LS74 clear
R20 1K
R21 1K
C7 0u1 (tant)
R18 1K
R15 15K
C5 100P
IC13a 74LS123
R16 15K
C6 100P
IC13b 74LS123
GND
LINE
FIELD
ZERO
+15v
LIGHT
SWITCH
TO PEN UNIT

FIG 15.

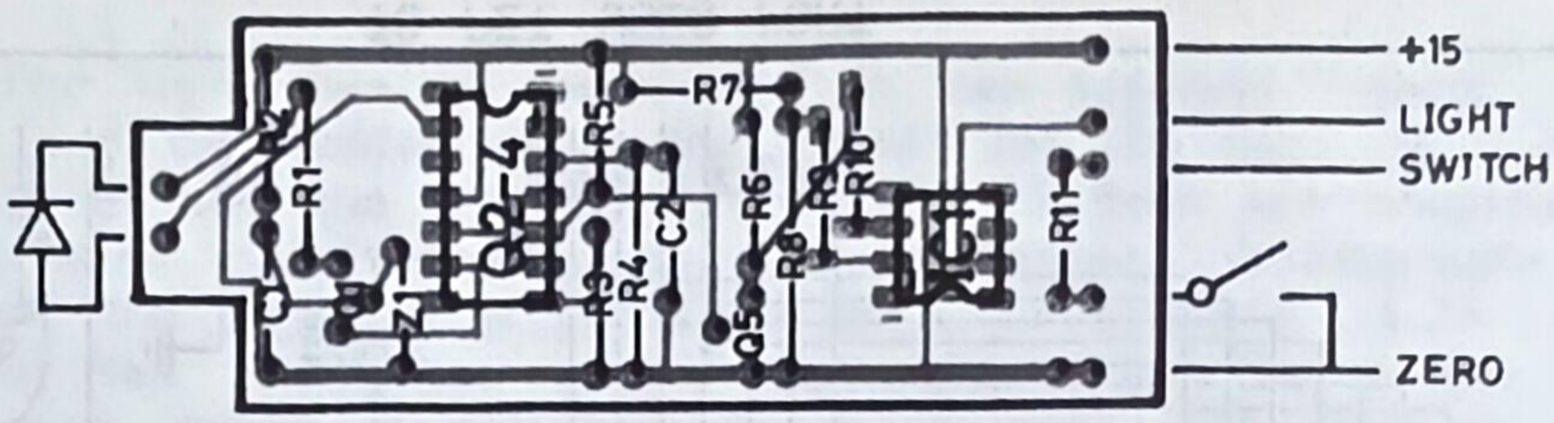
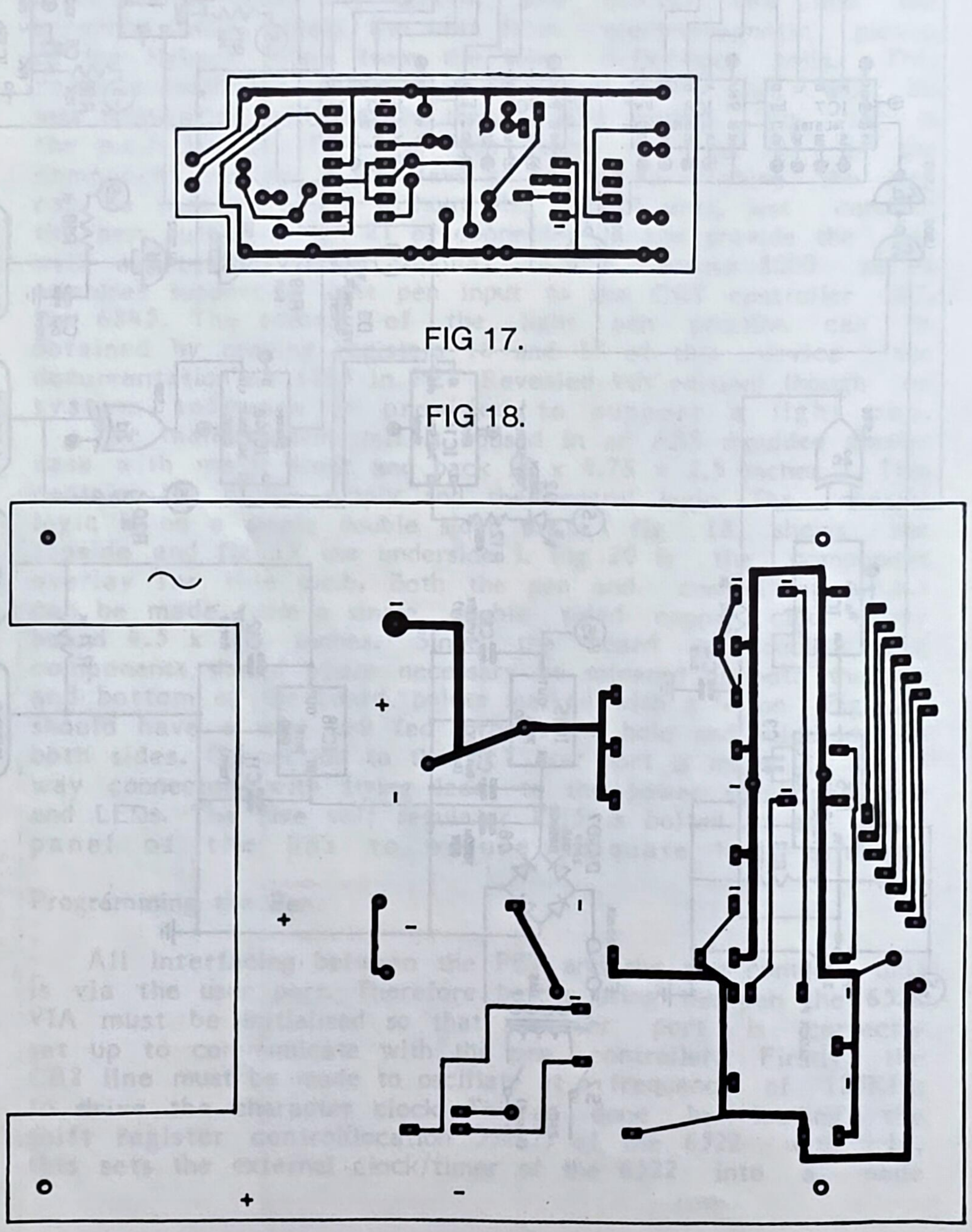


FIG 16.

FIG 17.

FIG 18.



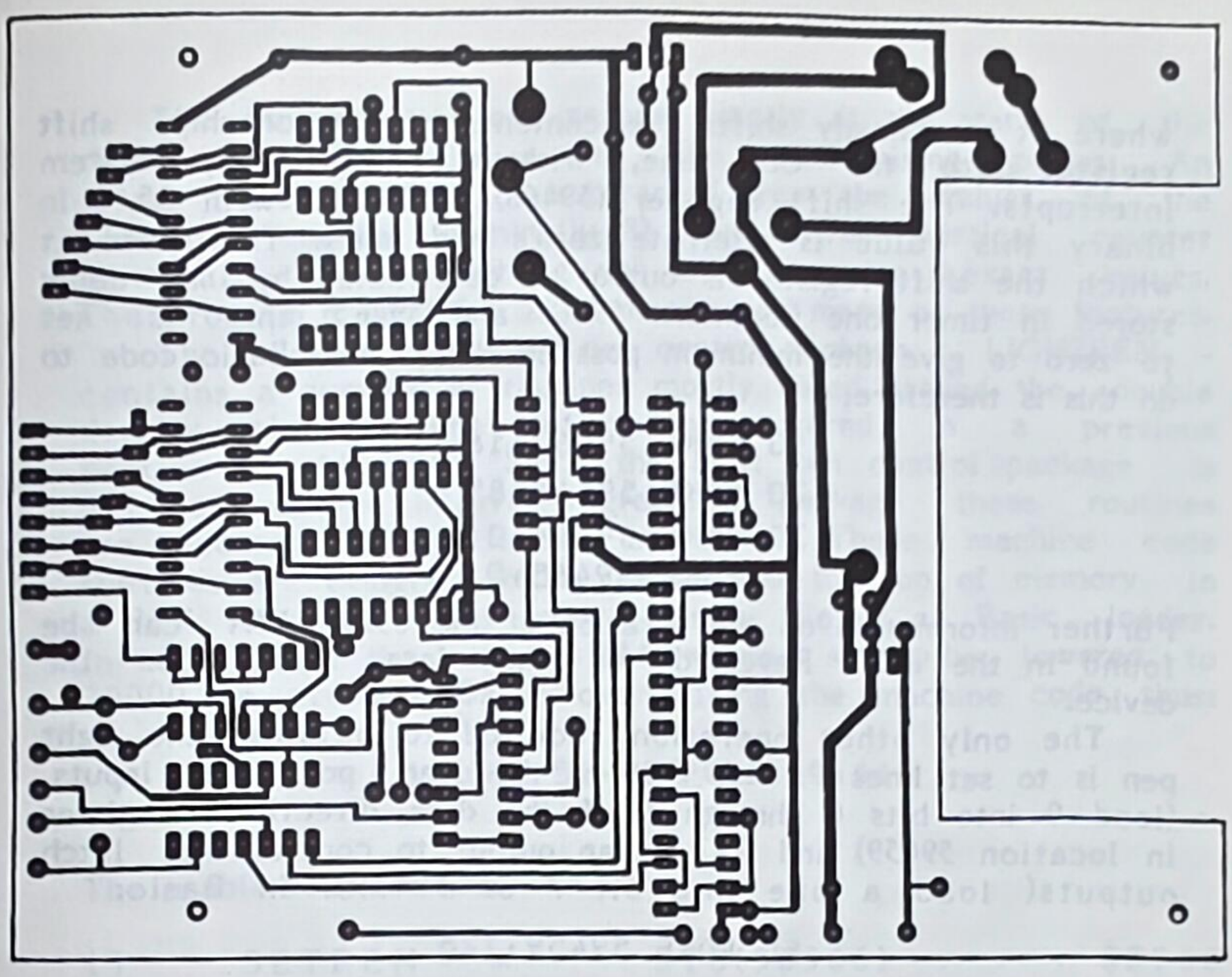
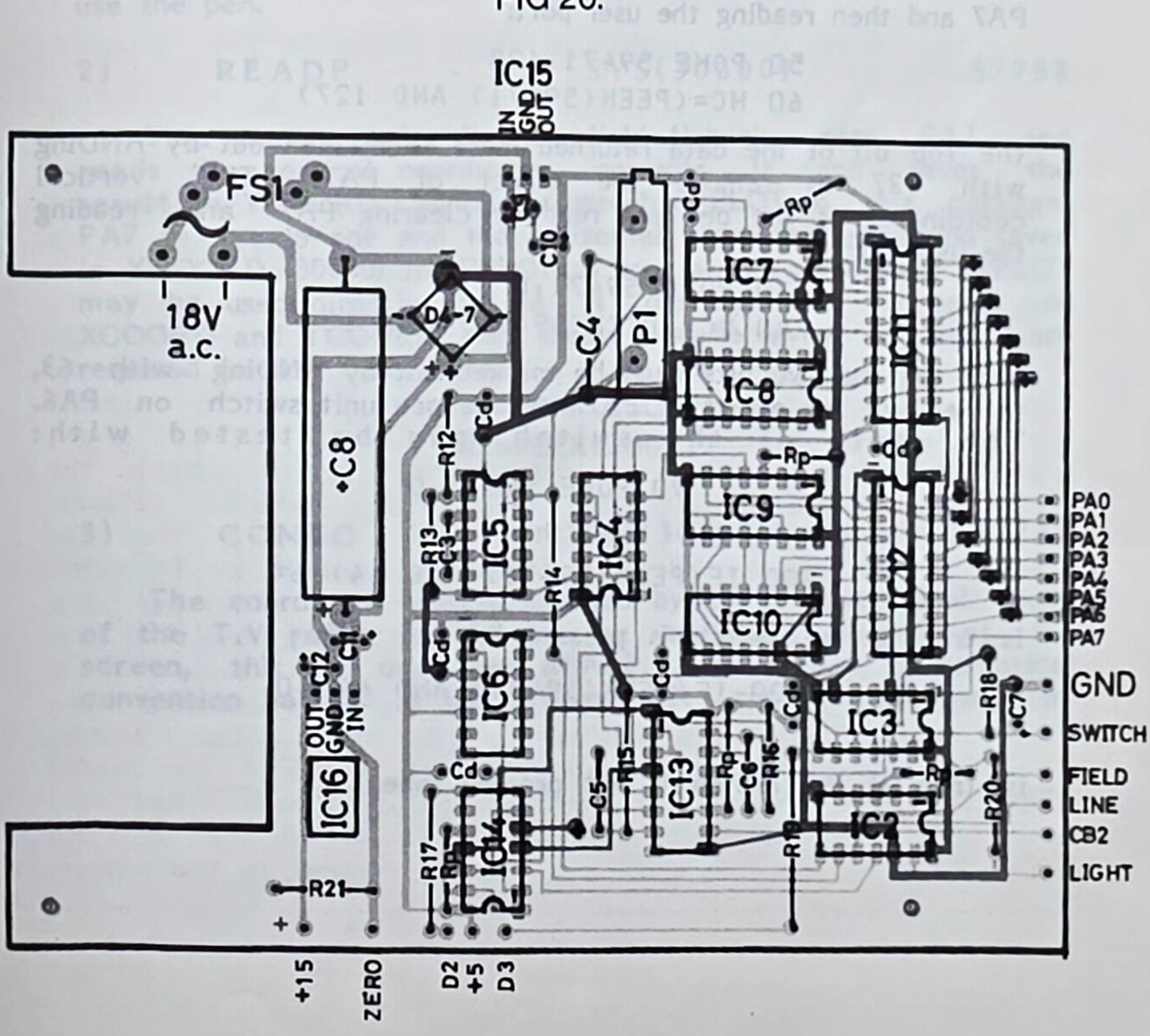


FIG 19.

FIG 20.


IC15
IN
GND
OUT
FS1
18V
a.c.
D4-7
C10
C4
P1
Cd
RP
IC7
IC8
IC9
IC10
IC11
IC12
Rp
C8
R12
R13
C3
IC5
IC4
R14
IC6
C5
R15
IC13
C6
R16
R19
IC3
IC2
R18
C7
R20
R1
IC16
C12
C11
OUT
GND
IN
R21
R17
IC14
+15
ZERO
D2
+5
D3
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
GND
SWITCH
FIELD
LINE
CB2
LIGHT

where it continualy shifts the contents of the on-chip shift register onto the CB2 line, without causing any system interrupts. The shift register (59466) is loaded with 85, in binary this value is alternate zero's and one's. The speed at which the shift register is output is determined by a delay stored in timer one (locations 59464 and 59465) this is set to zero to give the minimum possible delay. The Basic code to do this is therefore:

```
10 POKE 59467,16
20 POKE 59466,85
30 POKE 59464,0
40 POKE 59465,0
```

Further information on the opration of the 6522 VIA can be found in the 'PET Revealed' or the data sheets for the device.

The only other operation required to initialise the light pen is to set lines PA0 to PA6 of the user port as inputs, (load 0 into bits 0 through 6 of the data direction register in location 59459) and PA7 as an output to control the latch outputs(load a one into bit 7 of 59459). In Basic:

```
50 POKE 59459,128
```

SETPEN is an equivalent machine code routine, callable from Basic with SYS(30566) and part of the light pen control package. Once this has been done the pen may be used, the horizontal coordinates of the pen are read first by setting PA7 and then reading the user port:

```
50 POKE 59471,128
60 HC=(PEEK(59471) AND 127)
```

the top bit of the data returned must be masked out by ANDing with 127 to remove the effect of PA7. The vertical coordinates of the pen are read by clearing PA7 and reading the user port:

```
70 POKE 59471,0
80 VC=(PEEK(59471) AND 63)
```

here the top two bits must be masked out by ANDing with 63, to remove the effect of PA7 and the pen unit switch on PA6. The state of the switch may be tested with:

```
90 POKE 59471,0
```

then

```
100 IF(PEEK(59471) AND 64)=64
```

is true if the switch is pressed (closed) and:

```
100 IF(PEEK(59471) AND 64)=0
```

is true if the switch is not pressed (open).

The counters are not zeroed exactly at the start of the video signal in relation to the syncronising pulses. An offset must therefore be subtracted from the values of the horizontal counter (nominally 8) and the vertical counter (nominally 6), see 'setting up' to determine exact values. The program EXAMPLE 22 demonstrates many of these features.

The machine code light pen control package - LIGHTPEN - contains a number of routines mostly based around the double density point and line plot routines covered in a previous section of this book. Since the light pen control package is not part of the integrated graphics package these routines are located at different addresses. These machine code routines are designed to be loaded into the top of memory in a 32K machine, either from a binary file or a Basic loader. In either case the limit of RAM memory should be lowered to 30000 to prevent Basic' overwriting the machine code, thus:

```
1 POKE 52,48:POKE 53,117
```

The usable routines are:

1) SETPEN - SYS(30566) $7766

Initialises the user port for use with the light pen as previously described, it must be called (or equivalent operations performed in Basic) before any attempt is made to use the pen.

2) READP - SYS(30600) $7788

This routine waits for the field flyback, sets PA7 and reads the vertical coordinate, corrects it and saves the result in YCOORD (30001), as used by PPLOT in this package. PA7 is set to one and the horizontal component read and saved in XCOORD (30000) for PPLOT to use after correction. PPLOT may be used directly, but as it destroys the contents of XCOORD and YCOORD, they should first be saved if they are required for any other reason.

```
VC=PEEK(30000)
HC=PEEK(30001)
```

3) CONVC - SYS(30590) $777E

The coordinate system dictated by the conventional scan of the T.V raster has 0,0 in the top left corner of the screen, this is used by READP. However, mathematical convention has 0,0 in the bottom left corner, as used by

PPLOT and LPLOT. This routine converts the output of READP to the form used by PPLOT and LPLOT.

4) SAVES - SYS(30642) $77B2

This routine saves the 1K of memory mapped video display at $8000 to the last 1K of user RAM at $7C00, to be later restored using RESTRS (below).

5) RESTRS - SYS(30660) $77C4

Restores the 1K of video RAM saved at $7C00 by SAVES to the memory mapped video RAM at $8000. This routine and SAVES are used as a pair. Both SAVES and RESTRS call TRANSF at $77D6 to perform the actual movement of data.

6) BRIGHT - SYS(30699) $77EB

All bytes in the video display RAM are set to 160 by this routine, this is the reversed space character and consequently the whole screen goes bright.

7) RVSSC - SYS(30726) $7806

This routine reverses all the characters on the screen by performing a logical NOT on the top bit of each of the 1000 screen character locations (exclusive OR with one).

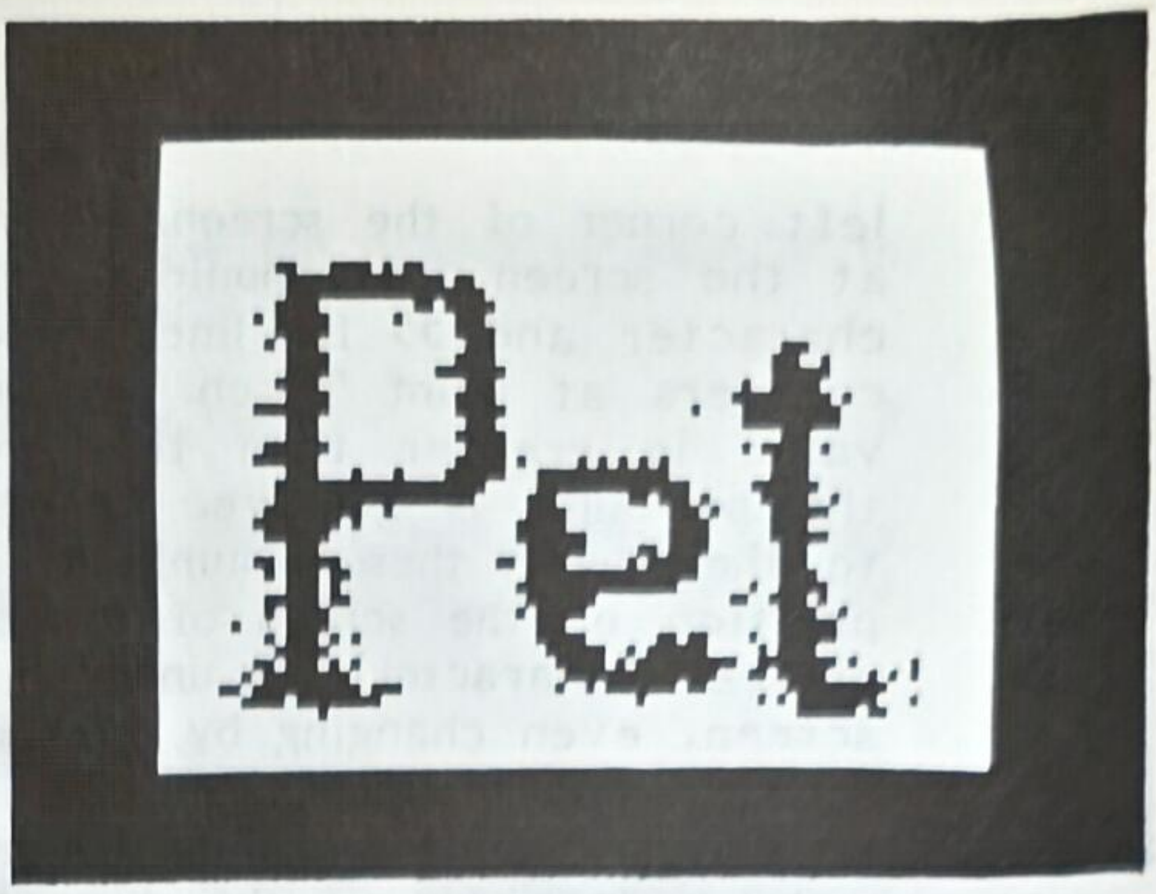
The use of all these routines is demonstrated in the program EXAMPLE 23

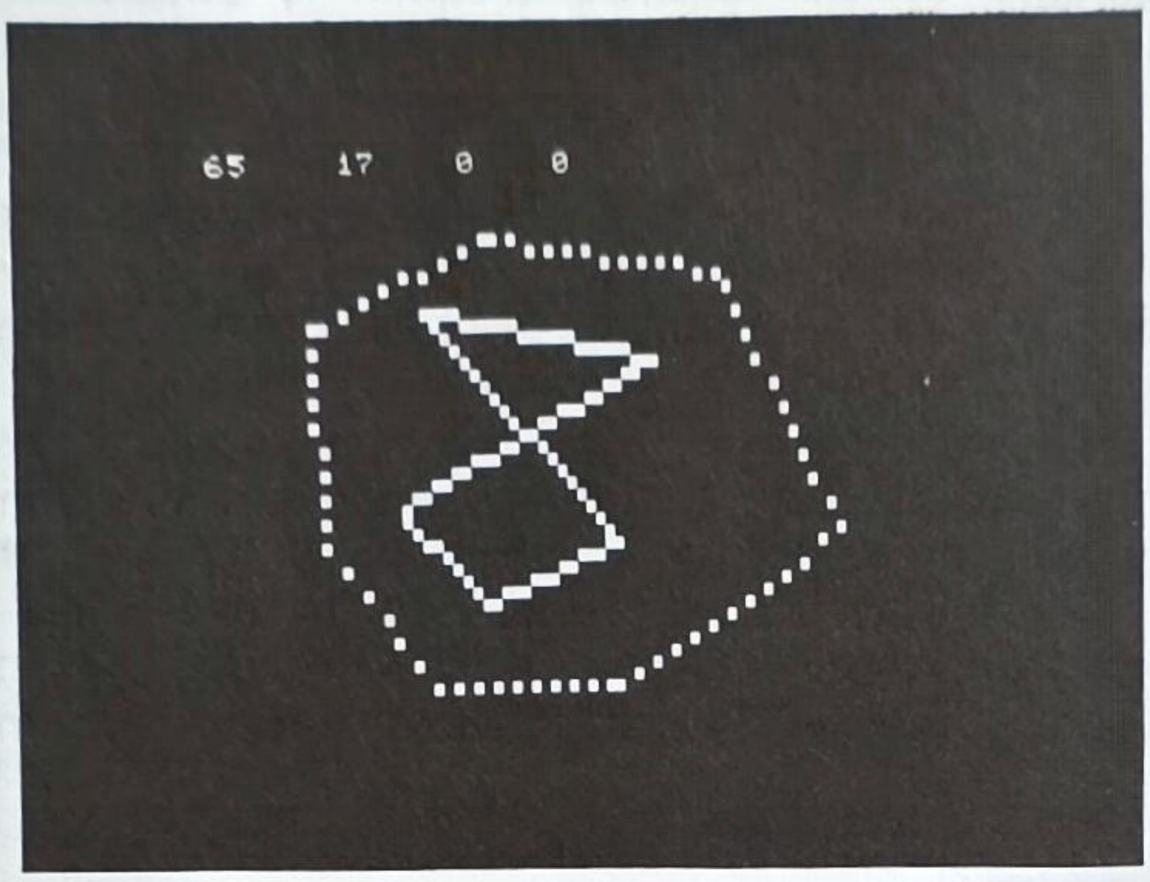
Setting up and notes for use.

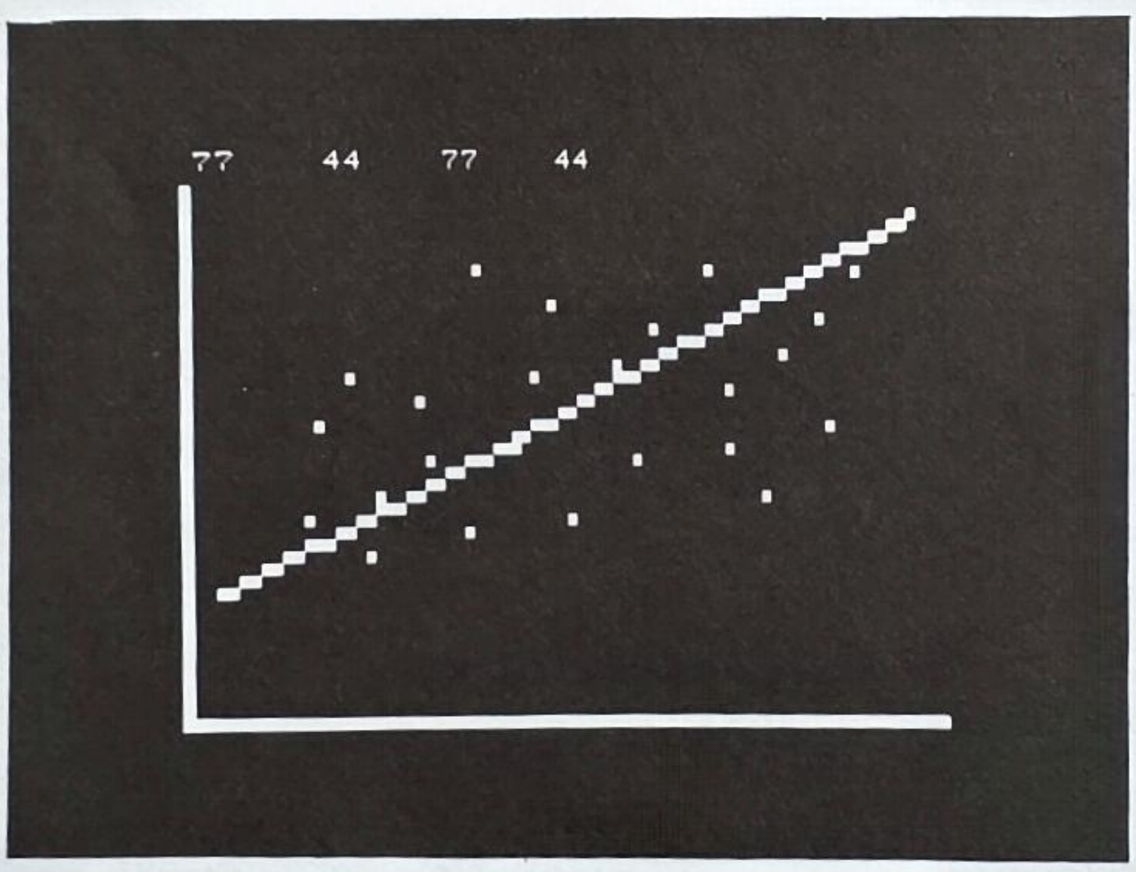
Once constructed the pen should be carefully checked and correctly connected to the PET user port. Setting up is a simple matter of adjusting the 'clock phase' preset P1, and determining the two values required to normalise the character and line scans to commence with zero. Switch on the pen unit and check that the yellow 'push' LED lights when the pen unit switch is pressed, and that the green 'on screen' LED lights when the pen is pointed directly at a bright portion of the screen. A number of factors affect the operation of the pen. Firstly, it may take several minutes for the head amplifier to settle and therefore the 'on screen' LED will be on for a while after switch on. Secondly, on the more common green screened PET the intensity of the screen (control at the back of the display) may have to be increased slightly. It should not have to be excessive, if brightness levels cause de-focusing of the characters the unit should be checked for faults. Once these initial checks have been completed the program EXAMPLE 24 should be loaded and run. This sets up the light pen as previously described and writes reverse spaces over the screen to bright it up. The value of the character and line are printed at the bottom

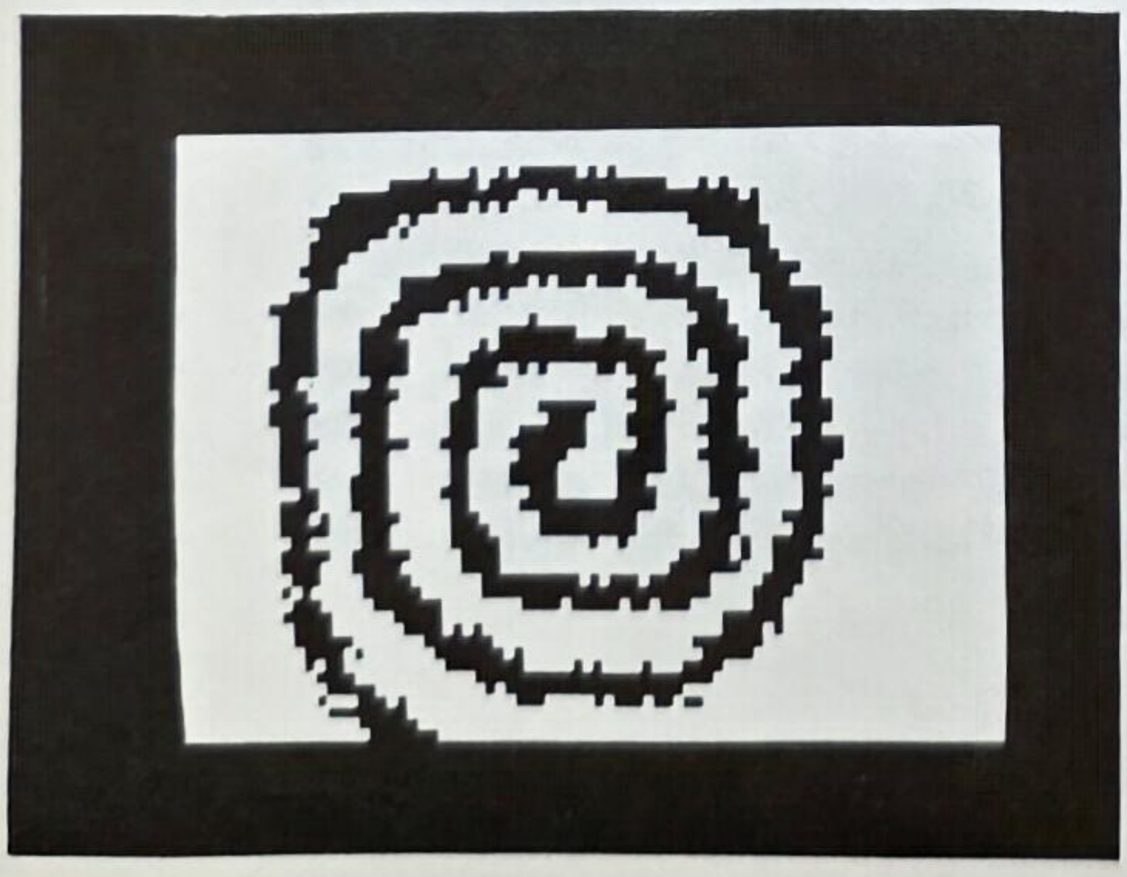
left corner of the screen. When the pen unit is not pointing at the screen they should be fixed at some value (i.e. 60 for character and 55 for line, these being the values for the counters at point 'C' on the timing diagram). As the line value is greater than that possible (49), this indicates that the pen unit is inactive. Whenever the pen is brought close to the screen these numbers should change indicating the position on the screen of the pen. If the value of the top number (character) is unstable when the pen is off the screen, even changing by one or two, then the phase locked loop is out of lock. The clock phase preset needs adjusting until this value is stable. Unfortunately the PLL may lock onto a harmonic of the desired frequency, check that this is not the case, by ensuring that the character readings with the pen at the far right of the screen is eighty greater than that from the far left of the screen. The lock at the fundamental is by far the strongest and when correctly set up there will be considerable travel of the preset where lock is retained. Set P1 to the centre of this range. Finally check that the top of the screen gives a line reading of zero, and the far left of the screen a character reading of zero(notwithstanding an unstable reading while 'on screen'). Offsets may be corrected by varying the values at the end of statements 220 for character and 240 for line. If these values differ from those given in EXAMPLE 24 then these values will have to be altered in any other program using the light pen.

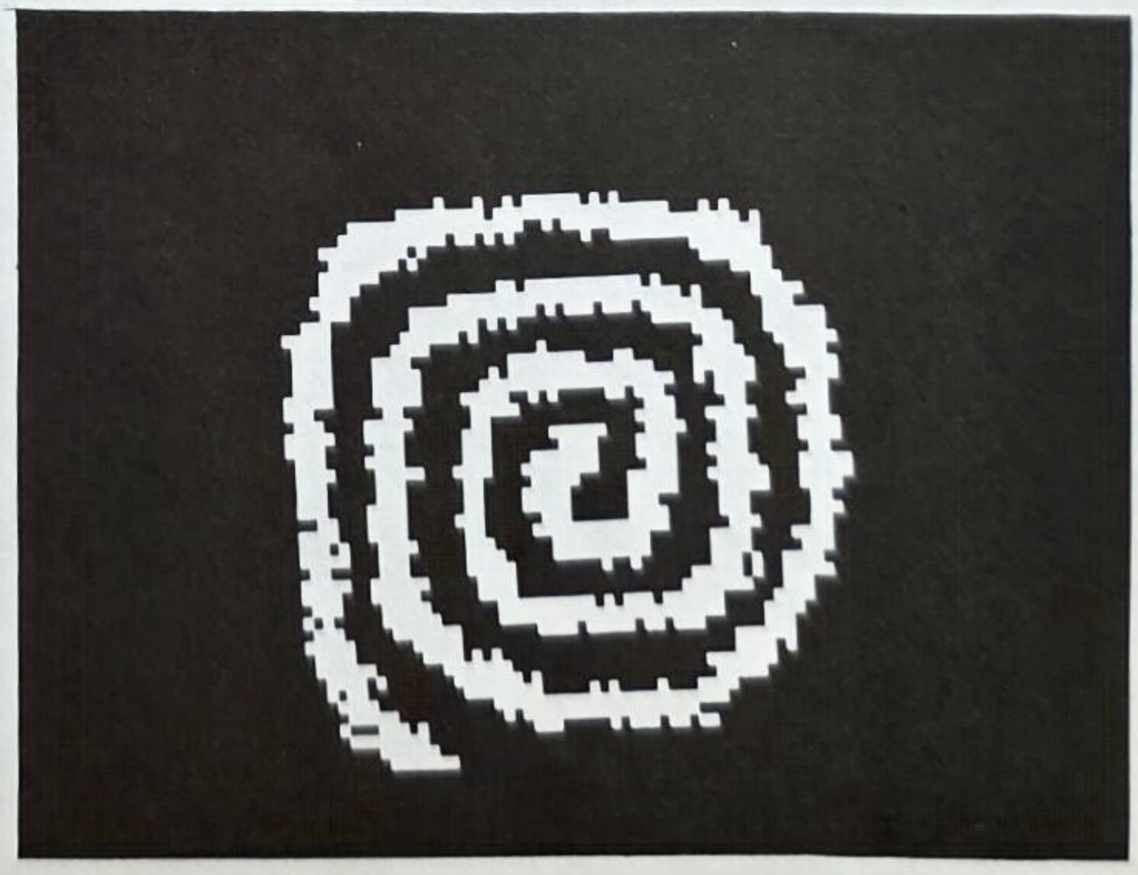
Pet


Pet


65 17 0 0


77 44 77 44





```
LINE# LOC    CODE          LINE

0409  7766                ;*************************
0410  7766                ;*                       *
0411  7766                ;*   PPPP  EEEEE N   N   *
0412  7766                ;*   P   P E     NN  N   *
0413  7766                ;*   P   P E     N N N   *
0414  7766                ;*   PPPP  EEEE  N N N   *
0415  7766                ;*   P     E     N N N   *
0416  7766                ;*   P     E     N  NN   *
0417  7766                ;*   P     EEEEE N   N   *
0418  7766                ;*                       *
0419  7766                ;*************************
0420  7766                ;
0421  7766                ;SYSTEM LOCATIONS
0422  7766                ;
0423  7766                TIM1LO=59464
0424  7766                TIM1HI=59465
0425  7766                SRCONT=59467
0426  7766                DDRA=59459
0427  7766                SHIFTR=59466
0428  7766                PEN=59471
0429  7766                ;
0430  7766                ;CONSTANTS
0431  7766                ;
0432  7766                XF=6
0433  7766                XL=8
0434  7766                RSPACE=160
0435  7766                ;
0436  7766                ;*******************************
0437  7766                ;*
0438  7766                ;*ROUTINE TO SET UP LIGHT PEN
0439  7766                ;*
0440  7766                ;*CB2 OUTPUT AT 125KHZ
0441  7766                ;*PA0-PA6 AS INPUTS
0442  7766                ;*PA7 AS OUTPUT
0443  7766                ;*
0444  7766                ;*******************************
0445  7766                ;
0446  7766  A9 10         SETPEN LDA #16
0447  7768  8D 4B E8             STA SRCONT
0448  776B  A9 55               LDA #85
0449  776D  8D 4A E8             STA SHIFTR
0450  7770  A9 00               LDA #0
0451  7772  8D 48 E8             STA TIM1LO
0452  7775  8D 49 E8             STA TIM1HI
0453  7778  A9 80               LDA #128
0454  777A  8D 43 E8             STA DDRA
0455  777D  60                   RTS
0456  777E                ;
0457  777E                ;*******************************
0458  777E                ;*
0459  777E                ;*ROUTINE TO CONVERT BETWEEN
0460  777E                ;*PEN COORDINATES (0,0)=TOP
0461  777E                ;*LEFT TO PPLOT COORDINATES
0462  777E                ;*(0,0)=BOTTOM LEFT.
0463  777E                ;*
```

```
LINE# LOC    CODE          LINE

0464  777E                ;*******************************
0465  777E                ;
0466  777E  A9 31         CONVC  LDA #49
0467  7780  38                   SEC
0468  7781  ED 31 75             SBC YCOORD
0469  7784  8D 31 75             STA YCOORD
0470  7787  60                   RTS
0471  7788                ;
0472  7788                ;*******************************
0473  7788                ;*
0474  7788                ;*ROUTINE TO READ IN PEN
0475  7788                ;*COORDINATES.   SAVED IN
0476  7788                ;*XCOORD & YCOORD READY FOR
0477  7788                ;*PPLOT, SAVE IF REQUIRED
0478  7788                ;*BEFORE CALLING PPLOT
0479  7788                ;*
0480  7788                ;*******************************
0481  7788                ;
0482  7788                ;WAIT FOR REFRESH
0483  7788                ;
0484  7788  AD 40 E8     READP  LDA REFRSH
0485  778B  49 20                EOR #$20
0486  778D  29 20                AND #$20
0487  778F  F0 F7                BEQ READP
0488  7791                ;SET PA7 TO ZERO
0489  7791  A9 00                LDA #0
0490  7793  8D 4F E8             STA PEN
0491  7796                ;READ FIELD
0492  7796  AD 4F E8             LDA PEN
0493  7799  29 3F                AND #63
0494  779B  38                   SEC
0495  779C  E9 06                SBC #XF
0496  779E  8D 31 75             STA YCOORD
0497  77A1                ;SET PA7 TO ONE
0498  77A1  A9 80                LDA #128
0499  77A3  8D 4F E8             STA PEN
0500  77A6                ;READ LINE
0501  77A6  AD 4F E8             LDA PEN
0502  77A9  29 7F                AND #127
0503  77AB  38                   SEC
0504  77AC  E9 08                SBC #XL
0505  77AE  8D 30 75             STA XCOORD
0506  77B1  60                   RTS
0507  77B2                ;
0508  77B2                ;*******************************
0509  77B2                ;*
0510  77B2                ;*ROUTINE TO SAVE CONTENTS OF
0511  77B2                ;*SCREEN IN RESERVED BLOCK OF
0512  77B2                ;*MEMORY
0513  77B2                ;*
0514  77B2                ;*******************************
0515  77B2                ;
0516  77B2                ;TEMP POINTER LOCATION
0517  77B2                YTEMP1=91
0518  77B2                ;
```

```
LINE# LOC    CODE          LINE

0519  77B2   A9 00      SAVES  LDA #0
0520  77B4   85 00             STA YTEMP
0521  77B6   85 5B             STA YTEMP1
0522  77B8   A9 80             LDA #$80
0523  77BA   85 01             STA YTEMP+1
0524  77BC   A9 7C             LDA #$7C
0525  77BE   85 5C             STA YTEMP1+1
0526  77C0   20 D6 77           JSR TRANSF
0527  77C3   60                 RTS
0528  77C4                ;
0529  77C4                ;*******************************
0530  77C4                ;*
0531  77C4                ;*ROUTINE TO RESTORE SCREEN FROM
0532  77C4                ;*RESERVED BLOCK OF MEMORY
0533  77C4                ;*
0534  77C4                ;*******************************
0535  77C4                ;
0536  77C4   A9 00      RESTRS LDA #0
0537  77C6   85 00             STA YTEMP
0538  77C8   85 5B             STA YTEMP1
0539  77CA   A9 7C             LDA #$7C
0540  77CC   85 01             STA YTEMP+1
0541  77CE   A9 80             LDA #$80
0542  77D0   85 5C             STA YTEMP1+1
0543  77D2   20 D6 77           JSR TRANSF
0544  77D5   60                 RTS
0545  77D6                ;
0546  77D6                ;*******************************
0547  77D6                ;*
0548  77D6                ;*TRANSFERS 1024 BYTES FROM
0549  77D6                ;*AREA POINTED TO BY YTEMP TO
0550  77D6                ;*AREA POINTED TO BY YTEMP1
0551  77D6                ;*
0552  77D6                ;*******************************
0553  77D6                ;
0554  77D6   A2 00      TRANSF LDX #0
0555  77D8   A0 00      LOOP2  LDY #0
0556  77DA   B1 00      LOOP1  LDA (YTEMP),Y
0557  77DC   91 5B             STA (YTEMP1),Y
0558  77DE   C8                 INY
0559  77DF   D0 F9             BNE LOOP1
0560  77E1   E6 01             INC YTEMP+1
0561  77E3   E6 5C             INC YTEMP1+1
0562  77E5   E8                 INX
0563  77E6   E0 04             CPX #4
0564  77E8   D0 EE             BNE LOOP2
0565  77EA   60                 RTS
0566  77EB                ;
0567  77EB                ;*******************************
0568  77EB                ;*
0569  77EB                ;*BRIGHTS UP SCREEN WITH
0570  77EB                ;*REVERSED SPACES
0571  77EB                ;*
0572  77EB                ;*******************************
0573  77EB                ;
```

```
LINE# LOC    CODE          LINE

0574  77EB   A9 00      BRIGHT LDA #0
0575  77ED   85 00             STA YTEMP
0576  77EF   A9 80             LDA #$80
0577  77F1   85 01             STA YTEMP+1
0578  77F3   A2 00             LDX #0
0579  77F5   A0 00       LP2   LDY #0
0580  77F7   A9 A0       LP1   LDA #RSPACE
0581  77F9   91 00             STA (YTEMP),Y
0582  77FB   C8                INY
0583  77FC   D0 F9             BNE LP1
0584  77FE   E6 01             INC YTEMP+1
0585  7800   E8                INX
0586  7801   E0 04             CPX #4
0587  7803   D0 F0             BNE LP2
0588  7805   60                RTS
0589  7806                   ;
0590  7806                   ;*********************************
0591  7806                   ;*
0592  7806                   ;*REVERSE ALL CHARACTERS ON SCREEN
0593  7806                   ;*
0594  7806                   ;*********************************
0595  7806                   ;
0596  7806   A9 00       RVSSC LDA #0
0597  7808   85 00             STA YTEMP
0598  780A   A9 80             LDA #$80
0599  780C   85 01             STA YTEMP+1
0600  780E   A2 00             LDX #0
0601  7810   A0 00       RV2   LDY #0
0602  7812   B1 00       RV1   LDA (YTEMP),Y
0603  7814   49 80             EOR #128
0604  7816   91 00             STA (YTEMP),Y
0605  7818   C8                INY
0606  7819   D0 F7             BNE RV1
0607  781B   E6 01             INC YTEMP+1
0608  781D   E8                INX
0609  781E   E0 04             CPX #4
0610  7820   D0 EE             BNE RV2
0611  7822   60                RTS
0612  7823                     .END
```

EXAMPLE 21

```
10 REM ************************
20 REM *
30 REM * PRINCIPLE OF LIGHT PEN
40 REM *
50 REM ************************
60 REM
70 REM SC=START OF SCREEN MEMORY
80 SC=8*4096
90 REM USER PORT INPUT, PHOTOCELL
100 REM CONNECTED TO PA0
110 POKE 59459,0
120 REM CLEAR SCREEN TO START
130 PRINT"□";
140 REM BRIGHT UP EACH CHARACTER IN
150 REM TURN
160 REM
170 FOR I=0 TO 24      :REM LINES
180 FOR J=0 TO 39      :REM CHARACTERS
190 POKE SC+(I*40)+J,160 :REM BRIGHT
200 REM WAIT
210 IF (PEEK(59471) AND 1)=1 THEN 330
220 POKE SC+(I*40)+J,32  :REM DARKEN
230 NEXT J
240 NEXT I
250 REM
260 REM PEN FAILS
270 REM
280 PRINT"PEN SAW NOTHING - NO COORDINATES"
290 STOP
300 REM
310 REM PEN SUCEEDS
320 REM
330 PRINT"PEN POINTING AT:"
340 PRINT"CHARACTER ";J;" ON LINE ";I
350 STOP
READY.
```

EXAMPLE 22

```
10 REM ************************************
20 REM *PROGRAM TO TEST OPERATION OF LIGHT PEN
30 REM ************************************
40 REM START CB2 AS CLOCK
50 POKE 59467,16
60 POKE 59466,85
70 POKE 59464,0
80 POKE 59465,0
90 REM PA0-PA6 INPUTS, PA7 AS OUTPUT
100 POKE 59459,128
110 REM WHITE UP SCREEN
```

```
120 FOR I=1 TO 100
130 PRINT"■                ■";
140 NEXT I
150 REM READ PIXEL AND LINE
160 POKE 59471,128
170 PX=(PEEK(59471) AND 127)-8
180 POKE 59471,0
190 LN=(PEEK(59471) AND 63)-6
200 REM PRINT VALUES
210 PRINT"■■■
```

```
370 PRINT"'R' - REVERSE THE SCREEN"
380 PRINT"'C' - CLEAR SCREEN TO START AGAIN"
390 PRINT"'P' - PRINT SCREEN ONTO PRINTER"
400 PRINT"'Q' - QUIT THIS PROGRAM"
410 GET A$
420 IF A$<>"" THEN 410
430 PRINT"  ** PRESS ANY KEY TO START **"
440 GET A$
450 IF A$="" THEN 440
460 REM SET UP PEN
470 SYS(30566)
480 REM BRIGHT SCREEN TO START
490 SYS(30699)
500 REM CHECK IF SWITCH PRESSED
510 POKE 59471,0
520 IF (PEEK(59471) AND 64)=0 THEN 600
530 REM READ PEN
540 SYS(30600)
550 REM CONVERT
560 SYS(30590)
570 REM ERASE
580 POKE 30002,1
590 SYS(30026)
600 REM CHECK KEYBOARD
610 GET A$
620 IF A$="" THEN 500
630 IF A$="R" THEN SYS(30726)
640 IF A$="C" THEN SYS(30699)
650 IF A$="P" THEN GOSUB 2920
660 IF A$="Q" THEN 130
670 GOTO 500
680 REM LIGHT PEN DRAW PROGRAM
690 PRINT"                    DRAW"
700 PRINT"
710 PRINT"A PROGRAM TO DRAW LINES ON THE SCREEN"
720 PRINT"USING LPLOT.  INITIAL AND FINAL"
730 PRINT"COORDINATES OF THE LINES MAY BE"
740 PRINT"CAPTURED USING THE PEN.  THE"
750 PRINT"DESTINATION COORDINATE MAY BE MOVED"
760 PRINT"AROUND USING THE DIGIT KEYS '0' - '9'."
770 PRINT"VARIOUS KEYS MAY BE USED TO ADD 'A',"
780 PRINT"ERASE 'E' OR DRAW DOTTED LINES 'D',"
790 PRINT"RESTART 'R' OR QUIT 'Q' THE PROGRAM OR"
800 PRINT"PRINT THE SCREEN 'P'.  IT IS OPTIONAL"
810 PRINT"AS TO WHETHER A POINT IS LEFT AS THE"
820 PRINT"CURSOR IS MOVED AROUND 'O', OR NOT 'F'."
830 GET A$
840 IF A$<>"" THEN 830
850 PRINT"  PRESS ANY KEY FOR COMMAND RESUME"
860 GET A$
870 IF A$="" THEN 860
880 PRINT" 'A' - ADD A LINE FROM X1,Y1 TO X2,Y2"
890 PRINT"'E' - ERASE A LINE BETWEEN POINTS"
900 PRINT"'D' - DRAW A DOTTED LINE X1,Y1 TO X2,Y2"
910 PRINT"'P' - PRINT SCREEN ONTO PRINTER"
920 PRINT"'R' - RESTART WITH A CLEAR SCREEN"
930 PRINT"'Q' - QUIT THIS SECTION OF PROGRAM"
940 PRINT"'O' - ON, CURSOR MOVEMENTS LEAVE POINTS"
950 PRINT"'F' - OFF, CANCELS 'O', (DEFAULT)"
960 PRINT"'M' - MOVE CURSOR ORIGIN"
```

```
970 PRINT"'1' - MOVE CURSOR DOWN AND LEFT"
980 PRINT"'2' - MOVE CURSOR DOWN"
990 PRINT"'3' - MOVE CURSOR DOWN AND RIGHT"
1000 PRINT"'4' - MOVE CURSOR LEFT"
1010 PRINT"'5' - LEAVE CURRENT POINT BRIGHT"
1020 PRINT"'6' - MOVE CURSOR RIGHT"
1030 PRINT"'7' - MOVE CURSOR UP AND LEFT"
1040 PRINT"'8' - MOVE CURSOR UP"
1050 PRINT"'9' - MOVE CURSOR UP AND RIGHT"
1060 PRINT
1070 PRINT"USE SWITCH ON PEN TO CAPTURE POINT"
1080 PRINT"RELEASE IMMEDIATLEY"
1090 GET A$
1100 IF A$<>"" THEN 1090
1110 PRINT"■■PRESS ANY KEY TO START"
1120 GET A$
1130 IF A$="" THEN 1120
1140 PRINT"■"
1150 X1=0
1160 Y1=0
1170 X2=0
1180 Y2=0
1190 DT=1
1200 TG=0
1210 TT=0
1220 PRINT"■";X1;"■ ";Y1;"■ ";X2;"■ ";Y2;"■ "
1230 REM BLINK CURSOR (X2,Y2)
1240 IF DT=1 THEN 1270
1250 DT=1
1260 GOTO 1280
1270 DT=0
1280 POKE 30000,X2
1290 POKE 30001,Y2
1300 POKE 30002,DT
1310 SYS(30026)
1320 REM READ KEYBOARD (ALSO TIMES CURSOR BLINK)
1330 GET A$
1340 IF A$="A" THEN GOSUB 1560
1350 IF A$="E" THEN GOSUB 1680
1360 IF A$="D" THEN GOSUB 1790
1370 IF A$="O" THEN GOSUB 1900
1380 IF A$="F" THEN GOSUB 1930
1390 IF A$="Q" THEN GOTO 130
1400 IF A$="R" THEN GOTO 1140
1410 IF A$="P" THEN GOSUB 1960
1420 IF A$="M" THEN GOSUB 2000
1430 IF A$="1" THEN GOSUB 2050
1440 IF A$="2" THEN GOSUB 2120
1450 IF A$="3" THEN GOSUB 2180
1460 IF A$="4" THEN GOSUB 2250
1470 IF A$="5" THEN GOSUB 2310
1480 IF A$="6" THEN GOSUB 2340
1490 IF A$="7" THEN GOSUB 2400
1500 IF A$="8" THEN GOSUB 2470
1510 IF A$="9" THEN GOSUB 2530
1520 REM READ PEN SWITCH
1530 POKE 59471,0
1540 IF (PEEK(59471) AND 64)=64 THEN GOSUB 2600
1550 GOTO 1220
1560 REM THE KEY SUBROUTINES
```

```
1570 REM 'A' DRAW A LINE
1580 POKE 30003,X1
1590 POKE 30004,Y1
1600 POKE 30005,X2
1610 POKE 30006,Y2
1620 POKE 30007,0
1630 SYS(30212)
1640 X1=X2
1650 Y1=Y2
1660 TT=0
1670 RETURN
1680 REM 'E' ERASE A LINE
1690 POKE 30003,X1
1700 POKE 30004,Y1
1710 POKE 30005,X2
1720 POKE 30006,Y2
1730 POKE 30007,1
1740 SYS(30212)
1750 X1=X2
1760 Y1=Y2
1770 TT=0
1780 RETURN
1790 REM 'D' DRAW A DOTTED LINE
1800 POKE 30003,X1
1810 POKE 30004,Y1
1820 POKE 30005,X2
1830 POKE 30006,Y2
1840 POKE 30007,2
1850 SYS(30212)
1860 X1=X2
1870 Y1=Y2
1880 TT=0
1890 RETURN
1900 REM 'O' DIGIT MOVE FLAG ON
1910 TG=1:TT=0
1920 RETURN
1930 REM 'F' DIGIT FLAG OFF
1940 TG=0:TT=0
1950 RETURN
1960 REM 'P' SCREEN TO PRINTER
1970 PRINT"■                                                    "
1980 GOSUB 2920
1990 RETURN
2000 REM 'M' MOVE CURSOR ORIGIN
2010 X1=X2
2020 Y1=Y2
2030 TT=0
2040 RETURN
2050 REM '1' CURSOR DOWN AND LEFT
2060 IF X2<1 OR Y2<1 THEN RETURN
2070 GOSUB 2850
2080 X2=X2-1
2090 Y2=Y2-1
2100 TT=0
2110 RETURN
2120 REM '2' CURSOR DOWN
2130 IF Y2<1 THEN RETURN
2140 GOSUB 2850
2150 Y2=Y2-1
2160 TT=0
```

```
2170 RETURN
2180 REM '3' CURSOR DOWN AND RIGHT
2190 IF Y2<1 OR X2>78 THEN RETURN
2200 GOSUB 2850
2210 X2=X2+1
2220 Y2=Y2-1
2230 TT=0
2240 RETURN
2250 REM '4' CURSOR LEFT
2260 IF X2<1 THEN RETURN>
2270 GOSUB 2850
2280 X2=X2-1
2290 TT=0
2300 RETURN
2310 REM '5' SAVE THIS POINT
2320 TT=1
2330 RETURN
2340 REM '6' CURSOR RIGHT
2350 IF X2>78 THEN RETURN
2360 GOSUB 2850
2370 X2=X2+1
2380 TT=0
2390 RETURN
2400 REM '7' CURSOR UP AND LEFT
2410 IF X2<1 OR Y2>46 THEN RETURN
2420 GOSUB 2850
2430 X2=X2-1
2440 Y2=Y2+1
2450 TT=0
2460 RETURN
2470 REM '8' CURSOR UP
2480 IF Y2>46 THEN RETURN
2490 GOSUB 2850
2500 Y2=Y2+1
2510 TT=0
2520 RETURN
2530 REM '9' CURSOR UP AND RIGHT
2540 IF X2>78 OR Y2>46 THEN RETURN
2550 GOSUB 2850
2560 X2=X2+1
2570 Y2=Y2+1
2580 TT=0
2590 RETURN
2600 REM COORDINATES FROM PEN, CLEAR CURRENT POINT?
2610 GOSUB 2850:TT=0
2620 REM SAVE SCREEN
2630 SYS(30642)
2640 REM BRIGHT SCREEN
2650 SYS(30699)
2660 REM GET PEN
2670 SYS(30600)
2680 REM CONVERT COORDINATES
2690 SYS(30590)
2700 REM RESTORE SCREEN
2710 SYS(30660)
2720 REM CHECK VALID RESULT
2730 XT=PEEK(30000)
2740 YT=PEEK(30001)
2750 IF YT<0 OR YT>49 THEN 2810
2760 Y2=YT
```

```
2770 X2=XT
2780 IF X2<0 THEN X2=0
2790 IF X2>78 THEN X2=79
2800 REM WAIT FOR SWITCH OFF
2810 POKE 59471,0
2820 IF (PEEK(59471) AND 64)=64 THEN 2820
2830 TT=0
2840 RETURN
2850 REM MOVE CURSOR
2860 POKE 30000,X2
2870 POKE 30001,Y2
2880 POKE 30002,1
2890 IF TG=1 OR TT=1 THEN POKE 30002,0
2900 SYS(30026)
2910 RETURN
2920 REM TRANSFER SCREEN TO PRINTER
2930 OPEN 4,4
2940 FOR I=32768 TO 33767 STEP 40
2950 A$=""
2960 RV=0
2970 K=0
2980 FOR J=I TO I+39
2990 CH=PEEK(J)
3000 IF CH<128 AND RV=1 THEN A$=A$+"■":RV=0
3010 IF CH>127 AND RV=0 THEN A$=A$+"■":RV=1
3020 IF CH>127 THEN CH=CH-128
3030 REM QUOTES SPECIAL CASE
3040 IF CH=34 AND K<39 THEN A$=A$+CHR$(34)+CHR$(34)+"■":GOTO 3080
3050 IF CH<32 OR CH>96 THEN A$=A$+CHR$(CH+64)
3060 IF CH>31 AND CH<64 THEN A$=A$+CHR$(CH)
3070 IF CH>63 AND CH<96 THEN A$=A$+CHR$(CH+128)
3080 K=K+1
3090 NEXT J
3100 PRINT#4,A$
3110 NEXT I
3120 CLOSE 4
3130 RETURN
READY.
```

EXAMPLE 24

```
10 REM ************************************
20 REM *PROGRAM EMULATES A POCKET CALCULATOR
30 REM *USE THE LIGHT PEN TO PRESS THE KEYS, JUST POINT
40 REM *THE PEN AT THE DESIRED KEY AND PRESS THE SWITCH
50 REM *ON THE PEN UNIT, THE RESULT WILL APPEAR IN THE
60 REM *DISPLAY AT THE TOP OF THE CALCULATOR.
70 REM ************************************
80 KY=1:REM 1=PRINT VERIFY, 0=SUPPRESSED
90 REM SET UP USER PORT FOR LIGHT PEN
100 POKE 59467,16
110 POKE 59466,85
120 POKE 59464,0
130 POKE 59465,0
140 POKE 59459,128
150 REM INITIALISE FLAGS
160 OF=4:REM OPERATOR FLAG
```

```
170 EF=0:REM ERROR FLAG
180 DP=0:REM DISPLAY REGISTER
190 BK=0:REM BACKING REGISTER
200 NO=1:REM NEW DISPLAY FLAG
210 S0=1:REM LEADING ZERO FLAG
220 PT=0:REM DECIMAL POINT FLAG
230 XF=6:REM FIELD OFFSET
240 XL=8:REM LINE OFFSET
250 REM SET UP GRAPHICS & ZERO DISPLAY
260 GOSUB 2530:GOSUB 2450:GOSUB 660
270 REM MAIN LOOP, WAIT FOR LIGHT
280 POKE 59471,0
290 F=(PEEK(59471) AND 127)-XF
300 IF F>42 THEN 290
310 REM READ POSITION ON SCREEN
320 POKE 59471,128
330 L=(PEEK(59471) AND 127)-XL
340 REM WHICH COLUMN
350 IF L>23 AND L<30 THEN LC=1:GOTO 400
360 IF L>33 AND L<40 THEN LC=2:GOTO 400
370 IF L>43 AND L<50 THEN LC=3:GOTO 400
380 IF L>53 AND L<60 THEN LC=4:GOTO 400
390 GOTO 270
400 REM WHICH ROW
410 IF F=18 OR F=19 THEN 480
420 IF F=22 OR F=23 THEN 500
430 IF F=26 OR F=27 THEN 520
440 IF F=30 OR F=31 THEN 540
450 IF F=34 OR F=35 THEN 560
460 IF F=38 OR F=39 THEN 580
470 GOTO 270
480 ON LC GOSUB 750,800,850,900
490 GOTO 590
500 ON LC GOSUB 950,1030,1110,1190
510 GOTO 590
520 ON LC GOSUB 1250,1300,1350,1400
530 GOTO 590
540 ON LC GOSUB 1460,1510,1560,1610
550 GOTO 590
560 ON LC GOSUB 1670,1720,1770,1820
570 GOTO 590
580 ON LC GOSUB 1880,1950,2010,2080
590 REM PRINT NEW DISPLAY
600 GOSUB 660
610 REM WAIT FOR LIGHT TO GO
620 POKE 59471,0
630 F=(PEEK(59471) AND 127)-XF
640 IF F<42 THEN 630
650 GOTO 270
660 REM PRINT NEW DISPLAY
670 IF EF=0 THEN 700
680 D$="                    E"
690 GOTO 710
700 D$=RIGHT$("                          "+D$,16)
710 PRINT"■■■
```

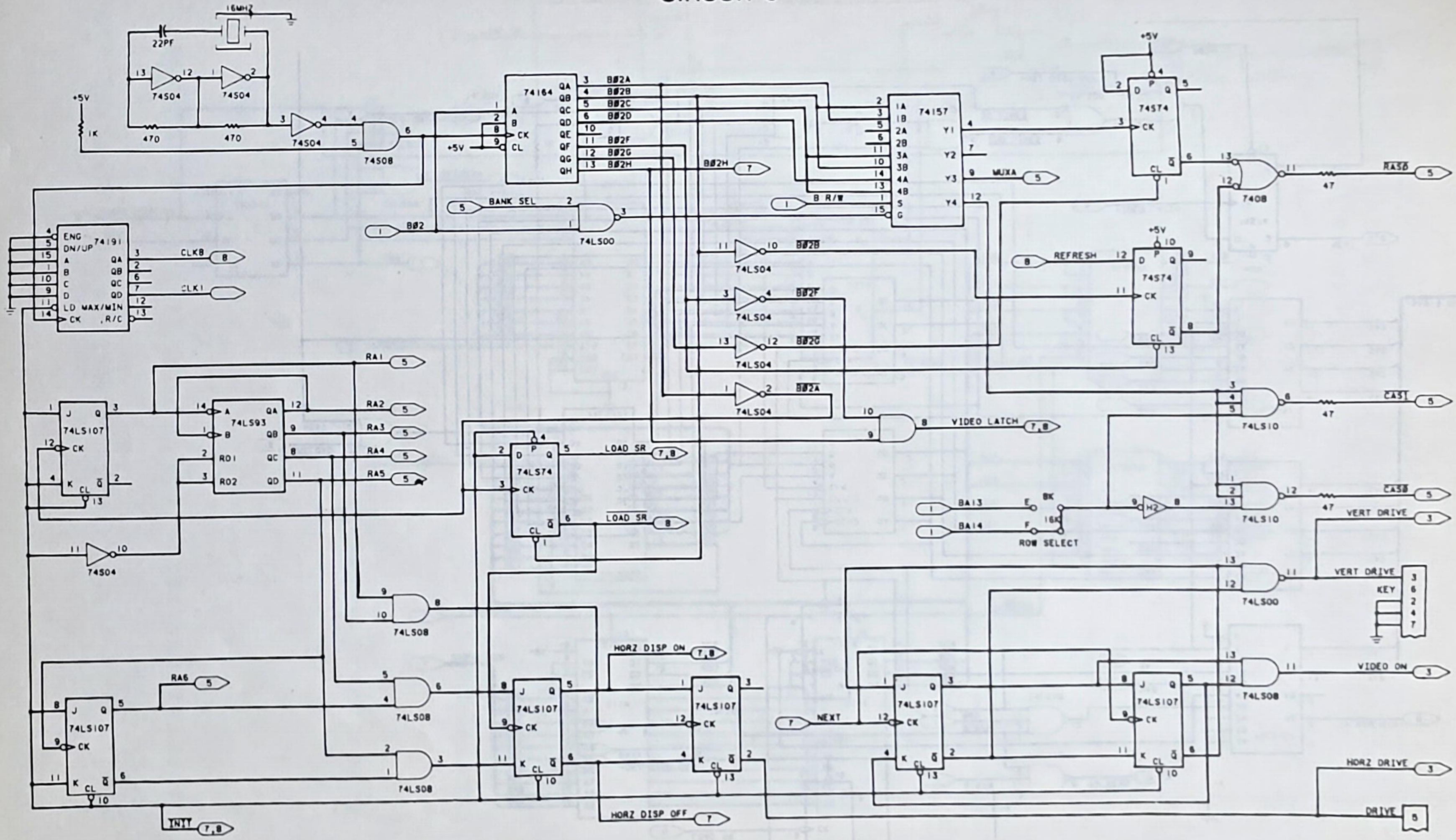
```
770 DP=SIN(DP/57.2957795)
780 GOSUB 2450
790 RETURN
800 REM COS()
810 IF KY THEN PRINT"■COS"
820 DP=COS(DP/57.2957795)
830 GOSUB 2450
840 RETURN
850 REM TAN()
860 IF KY THEN PRINT"■TAN"
870 DP=TAN(DP/57.2957795)
880 GOSUB 2450
890 RETURN
900 REM CE (CLEAR THIS ENTRY)
910 IF KY THEN PRINT"■C/E"
920 DP=0
930 GOSUB 2450
940 RETURN
950 REM SQR() (SQUARE ROOT)
960 IF KY THEN PRINT"■SQR"
970 IF DP >=0 THEN 1000
980 EF=1
990 RETURN
1000 DP=SQR(DP)
1010 GOSUB 2450
1020 RETURN
1030 REM LOG() (LOG BASE 10)
1040 IF KY THEN PRINT"■LOG"
1050 IF DP >0 THEN 1080
1060 EF=1
1070 RETURN
1080 DP=LOG(DP)/2.30258509
1090 GOSUB 2450
1100 RETURN
1110 REM 1/X() (RECIPRICAL)
1120 IF KY THEN PRINT"■1/X"
1130 IF DP<>0 THEN 1160
1140 EF=1
1150 RETURN
1160 DP=1/DP
1170 GOSUB 2450
1180 RETURN
1190 REM * (MULTIPLY)
1200 IF KY THEN PRINT"■ * "
1210 BK=DP
1220 OF=1
1230 GOSUB 2450
1240 RETURN
1250 REM DIGIT '7'
1260 IF KY THEN PRINT"■ 7 "
1270 DG$="7"
1280 GOSUB 2320
1290 RETURN
1300 REM DIGIT '8'
1310 IF KY THEN PRINT"■ 8 "
1320 DG$="8"
1330 GOSUB 2320
1340 RETURN
1350 REM DIGIT '9'
1360 IF KY THEN PRINT"■ 9 "
```

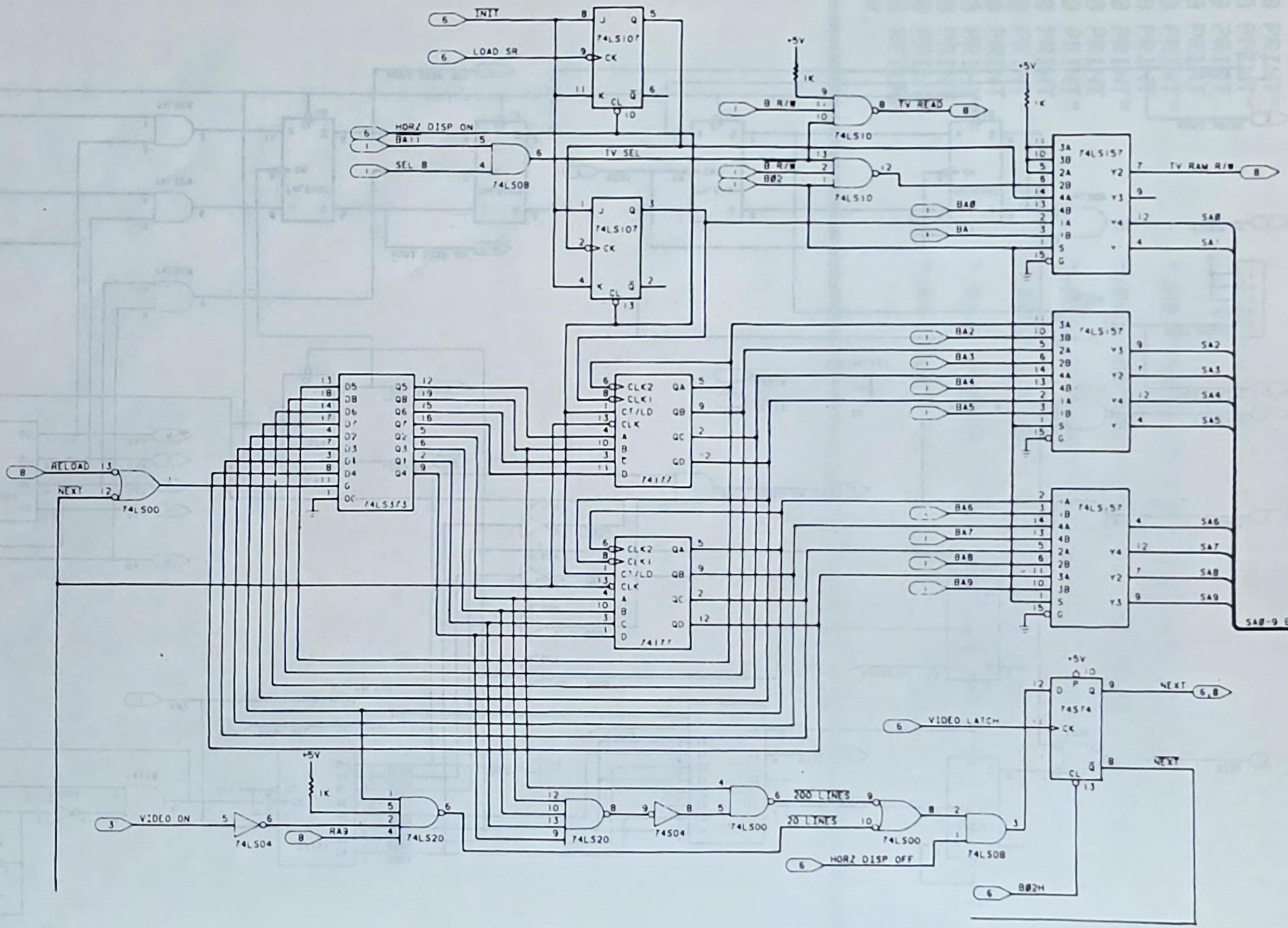
```
1370 DG$="9"
1380 GOSUB 2320
1390 RETURN
1400 REM / (DIVIDE)
1410 IF KY THEN PRINT"■ / "
1420 BK=DP
1430 OF=2
1440 GOSUB 2450
1450 RETURN
1460 REM DIGIT '4'
1470 IF KY THEN PRINT"■ 4 "
1480 DG$="4"
1490 GOSUB 2320
1500 RETURN
1510 REM DIGIT '5'
1520 IF KY THEN PRINT"■ 5 "
1530 DG$="5"
1540 GOSUB 2320
1550 RETURN
1560 REM DIGIT '6'
1570 IF KY THEN PRINT"■ 6 "
1580 DG$="6"
1590 GOSUB 2320
1600 RETURN
1610 REM - (SUBTRACT)
1620 IF KY THEN PRINT"■ - "
1630 BK=DP
1640 OF=3
1650 GOSUB 2450
1660 RETURN
1670 REM DIGIT '1'
1680 IF KY THEN PRINT"■ 1 "
1690 DG$="1"
1700 GOSUB 2320
1710 RETURN
1720 REM DIGIT '2'
1730 IF KY THEN PRINT"■ 2 "
1740 DG$="2"
1750 GOSUB 2320
1760 RETURN
1770 REM DIGIT '3'
1780 IF KY THEN PRINT"■ 3 "
1790 DG$="3"
1800 GOSUB 2320
1810 RETURN
1820 REM + (ADD)
1830 IF KY THEN PRINT"■ + "
1840 OF=4
1850 BK=DP
1860 GOSUB 2450
1870 RETURN
1880 REM CLR (CLEAR ALL)
1890 IF KY THEN PRINT"■CLR"
1900 DP=0
1910 BK=0
1920 OF=4
1930 GOSUB 2450
1940 RETURN
1950 REM DIGIT "0"
1960 IF KY THEN PRINT"■ 0 "
```

```
1970 IF S0=1 THEN RETURN
1980 DG$="0"
1990 GOSUB 2320
2000 RETURN
2010 REM '.' (DECIMAL POINT)
2020 IF KY THEN PRINT"■ . "
2030 IF PT=1 THEN RETURN
2040 DG$="."
2050 PT=1
2060 GOSUB 2320
2070 RETURN
2080 REM '=' (DO ARITHMETIC)
2090 IF KY THEN PRINT"■ = "
2100 ON OF GOTO 2130,2170,2240,2280
2110 PRINT"OPERATOR ERROR"
2120 STOP
2130 REM MULTIPLY
2140 DP=DP*BK
2150 GOSUB 2450
2160 RETURN
2170 REM DIVIDE
2180 IF DP<>0 THEN 2210
2190 EF=1
2200 RETURN
2210 DP=BK/DP
2220 GOSUB 2450
2230 RETURN
2240 REM SUBTRACT
2250 DP=BK-DP
2260 GOSUB 2450
2270 RETURN
2280 REM ADD
2290 DP=BK+DP
2300 GOSUB 2450
2310 RETURN
2320 REM ADD NEW DIGIT TO DP
2330 IF ND<>1 THEN 2360
2340 REM NEW DISPLAY
2350 ND=0
2360 IF LEN(D1$+D2$)>8+PD THEN RETURN
2370 IF PD=1 THEN 2400
2380 D1$=D1$+DG$
2390 GOTO 2410
2400 D2$=D2$+DG$
2410 D$=D1$+D2$
2420 DP=VAL(D$)
2430 S0=0
2440 RETURN
2450 REM SET FLAGS FOR NEW DISPLAY
2460 ND=1:REM NEW DISPLAY ON NEXT DIGIT
2470 S0=1:REM SUPPRESS ZEROS
2480 PT=0:REM NO . IN DISPLAY UNTIL PT=1
2490 D1$=""
2500 D2$=""
2510 D$=STR$(DP)
2520 RETURN
2530 REM PRINT OUT THE KEYPAD ON SCREEN
2540 R$="■   ■"
2550 PRINT"┌┬┬┬┐"
2560 PRINT"                    └──────────────────────┘"
```

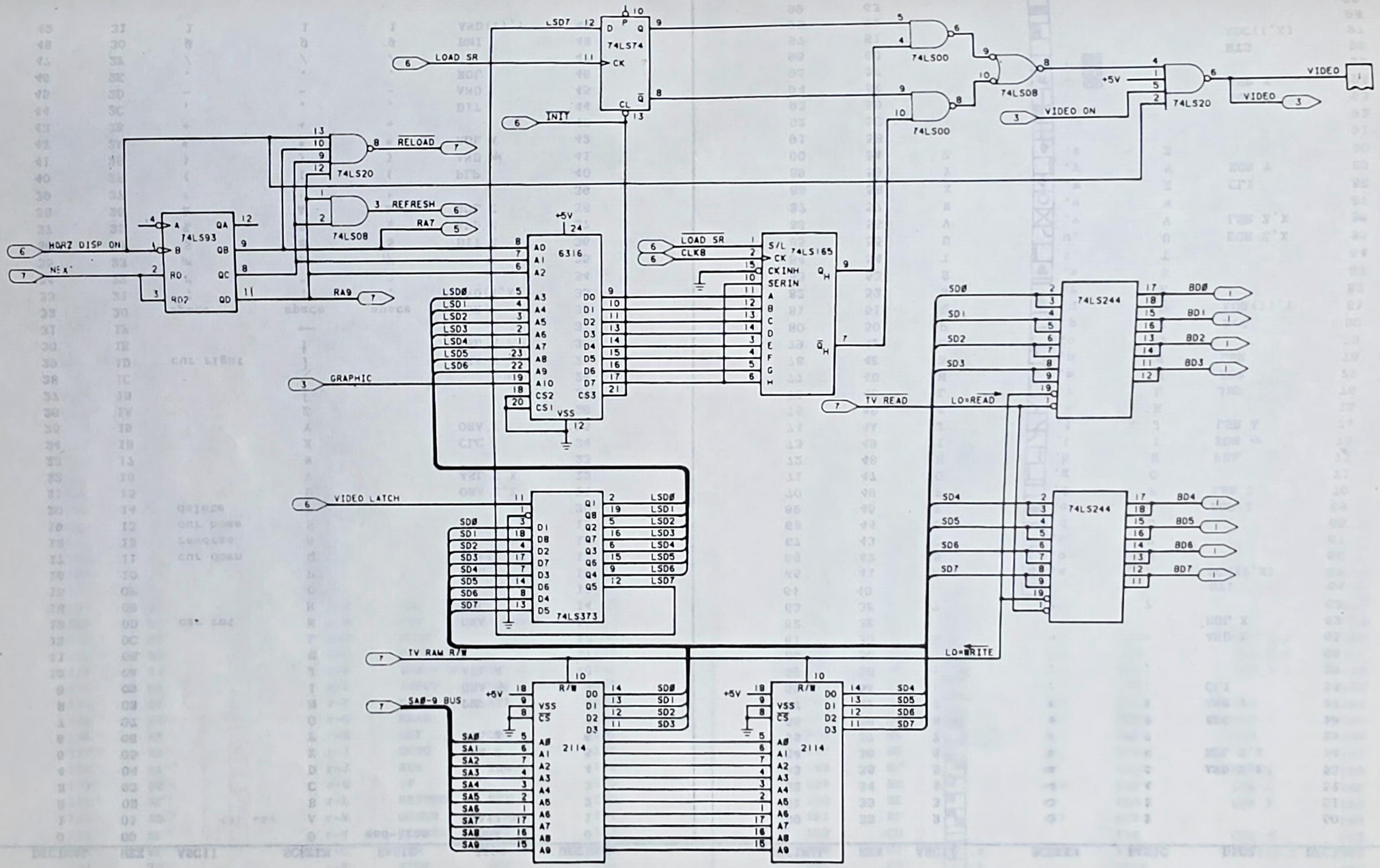
```
2570 PRINT"                    |                              |"
2580 PRINT"                    \______________________________/"
2590 PRINT
2600 PRINT
2610 PRINT"                ";R$;"SIN";R$;"COS";R$;"TAN";R$;"C/E"
2620 PRINT
2630 PRINT"                ";R$;"SQR";R$;"LOG";R$;"1/X";R$;" * "
2640 PRINT
2650 PRINT"                ";R$;" 7 ";R$;" 8 ";R$;" 9 ";R$;" / "
2660 PRINT
2670 PRINT"                ";R$;" 4 ";R$;" 5 ";R$;" 6 ";R$;" - "
2680 PRINT
2690 PRINT"                ";R$;" 1 ";R$;" 2 ";R$;" 3 ";R$;" + "
2700 PRINT
2710 PRINT"                ";R$;"CLR";R$;" 0 ";R$;" . ";R$;" = "
2720 RETURN
READY.
```

CIRCUIT 6


16MHZ
22PF
74S04
74S04
74S04
74S08
1K
470
470
74164
74LS00
BANK SEL
BØ2
BØ2H
74157
B R/W
MUXA
74LS04
BØ2B
BØ2F
BØ2G
BØ2A
74LS10
VIDEO LATCH
74S74
REFRESH
RASØ
CASØ
CAS1
47
7408
74191
ENG
DN/UP
LD MAX/MIN
CK ,R/C
CLKØ
CLK1
74LS107
74LS93
RA1
RA2
RA3
RA4
RA5
RA6
74LS08
74LS74
LOAD SR
LOAD SR
HORZ DISP ON
HORZ DISP OFF
INIT
BA13
BA14
ROW SELECT
16K
H2
VERT DRIVE
KEY
3 6 2 4 7
VIDEO ON
HORZ DRIVE
DRIVE
NEXT


INIT
LOAD SR
74LS107
CK
CL
HORZ DISP ON
BA11
SEL B
74LS08
TV SEL
+5V
1K
B R/W
B TV READ
74LS10
B R/W
B02
74LS157
BA0
BA
TV RAM R/W
SA0
SA1
BA2
BA3
BA4
BA5
SA2
SA3
SA4
SA5
BA6
BA7
BA8
BA9
SA6
SA7
SA8
SA9
SA0-9 BUS
RELOAD
NEXT
74LS00
74LS373
CLK2
CLK1
CT/LD
CLK
74177
VIDEO ON
74LS04
RA9
74LS20
200 LINES
240 LINES
74504
74LS08
HORZ DISP OFF
B02H
74S74
VIDEO LATCH
NEXT

CIRCUIT 7


74LS74
LOAD SR
INIT
LSD7
74LS00
74LS08
74LS20
VIDEO
VIDEO ON
+5V
74LS93
HORZ DISP ON
RELOAD
REFRESH
RA7
RA9
6316
GRAPHIC
LSD0
LSD1
LSD2
LSD3
LSD4
LSD5
LSD6
74LS165
CLK8
74LS244
SD0
SD1
SD2
SD3
SD4
SD5
SD6
SD7
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
TV READ
LO=READ
LO=WRITE
VIDEO LATCH
74LS373
TV RAM R/W
SA8-9 BUS
2114

CIRCUIT 8

Appendix 2

Table of PET Codes

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
0	00		@	end-line	BRK	0
1	01		A		ORA(I,X)	1
2	02		B			2
3	03		C			3
4	04		D			4
5	05		E		ORA Z	5
6	06		F		ASL Z	6
7	07		G			7
8	08		H		PHP	8
9	09		I		ORA #	9
10	0A		J		ASL A	10
11	0B		K			11
12	0C		L			12
13	0D	car ret	M		ORA	13
14	0E		N		ASL	14
15	0F		O			15
16	10		P		BPL	16
17	11	cur down	Q		ORA(I),Y	17
18	12	reverse	R			18
19	13	cur home	S			19
20	14	delete	T			20
21	15		U		ORA Z,X	21
22	16		V		ASL Z,X	22
23	17		W			23
24	18		X		CLC	24
25	19		Y		ORA Y	25
26	1A		Z			26
27	1B		[			27
28	1C		\			28
29	1D	cur right	]		ORA X	29
30	1E		↑		ASL X	30
31	1F		←			31
32	20	space	space	space	JSR	32
33	21	!	!	!	AND(I,X)	33
34	22	"	"	"		34
35	23	#	#	#		35
36	24	$	$	$	BIT Z	36
37	25	%	%	%	AND Z	37
38	26	&	&	&	ROL Z	38
39	27	'	'	'		39
40	28	(	(	(	PLP	40
41	29	)	)	)	AND #	41
42	2A	*	*	*	ROL A	42
43	2B	+	+	+		43
44	2C	,	,	,	BIT	44
45	2D	-	-	-	AND	45
46	2E	.	.	.	ROL	46
47	2F	/	/	/		47
48	30	0	0	0	BMI	48
49	31	1	1	1	AND(I),Y	49

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
50	32	2	2	2		50
51	33	3	3	3		51
52	34	4	4	4		52
53	35	5	5	5	AND Z,X	53
54	36	6	6	6	ROL Z,X	54
55	37	7	7	7		55
56	38	8	8	8	SEC	56
57	39	9	9	9	AND Y	57
58	3A	:	:	:	CLI	58
59	3B	;	;	;		59
60	3C					60
61	3D	=	=	=	AND X	61
62	3E				ROL X	62
63	3F	?	?	?		63
64	40				RTI	64
65	41	A	,a	A	EOR(I,X)	65
66	42	B	,b	B		66
67	43	C	,c	C		67
68	44	D	,d	D		68
69	45	E	,e	E	EOR Z	69
70	46	F	,f	F	LSR Z	70
71	47	G	,g	G		71
72	48	H	,h	H	PHA	72
73	49	I	,i	I	EOR #	73
74	4A	J	,j	J	LSR A	74
75	4B	K	,k	K		75
76	4C	L	,l	L	JMP	76
77	4D	M	,m	M	EOR	77
78	4E	N	,n	N	LSR	78
79	4F	O	,o	O		79
80	50	P	,p	P	BVC	80
81	51	Q	,q	Q	EOR(I),Y	81
82	52	R	,r	R		82
83	53	S	,s	S		83
84	54	T	t,	T		84
85	55	U	u,	U	EOR Z,X	85
86	56	V	,v	V	LSR Z,X	86
87	57	W	,w	W		87
88	58	X	,x	X	CLI	88
89	59	Y	,y	Y	EOR Y	89
90	5A	Z	,z	Z		90
91	5B					91
92	5C					92
93	5D				EOR X	93
94	5E				LSR X	94
95	5F					95
96	60				RTS	96
97	61				ADC(I,X)	97
98	62					98
99	63					99

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
100	64					100
101	65				ADC Z	101
102	66				RCR Z	102
103	67					103
104	68				PLA	104
105	69				ADC #	105
106	6A				RCR A	106
107	6B					107
108	6C				JMP(I)	108
109	6D				ADC	109
110	6E				ROR	110
111	6F					111
112	70				BVS	112
113	71				ADC(I),Y	113
114	72					114
115	73					115
116	74					116
117	75				ADC Z,X	117
118	76				ROR Z,X	118
119	77					119
120	78				SEI	120
121	79				ADC Y	121
122	7A					122
123	7B					123
124	7C					124
125	7D				ADC X	125
126	7E				ROR X	126
127	7F					127
128	80		r-O	END		128
129	81		r-A	FOR	STA(I,X)	129
130	82		r-B	NEXT		130
131	83		r-C	DATA		131
132	84		r-D	INPUT #	STY Z	132
133	85		r-E	INPUT	STA Z	133
134	86		r-F	DIM	STX Z	134
135	87		r-G	READ		135
136	88		r-H	LET	DEY	136
137	89		r-I	GOTO		137
138	8A		r-J	RUN	TXA	138
139	8B		r-K	IF		139
140	8C		r-L	RESTORE	STY	140
141	8D	car ret	r-M	GOSUB	STA	141
142	8E		r-N	RETURN	STX	142
143	8F		r-O	REM		143
144	90		r-P	STOP	BCC	144
145	91	cur up	r-Q	ON	STA(I),Y	145
146	92	rvs off	r-R	WAIT		146
147	93	clear	r-S	LOAD		147
148	94	insert	r-T	SAVE	STY Z,X	148
149	95		r-U	VERIFY	STA Z,X	149

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
150	96		r-V	DEF	STX Z,Y	150
151	97		r-W	POKE		151
152	98		r-X	PRINT #	TYA	152
153	99		r-Y	PRINT	STA Y	153
154	9A		r-Z	CONT	TXS	154
155	9B		r-[	LIST		155
156	9C		r-\	CLR		156
157	9D	cur left	r-]	CMD	STA X	157
158	9E		r-↑	SYS		158
159	9F		r-←	OPEN		159
160	A0			CLOSE	LDY #	160
161	A1		r-!	GET	LDA(I,X)	161
162	A2		r-"	NEW	LDX #	162
163	A3		r-#	TAB(		163
164	A4		r-$	TO	LDY Z	164
165	A5		r-%	FN	LDA Z	165
166	A6		r-&	SPC(	LDX Z	166
167	A7		r-'	THEN		167
168	A8		r-(	NOT	TAY	168
169	A9		r-)	STEP	LDA #	169
170	AA		r-*	+	TAX	170
171	AB		r-+	-		171
172	AC		r-,	*	LDY	172
173	AD		r--	/	LDA	173
174	AE		r-.		LDX	174
175	AF		r-/	AND		175
176	B0		r-0	OR	BCS	176
177	B1		r-1		LDA(I),Y	177
178	B2		r-2	=		178
179	B3		r-3			179
180	B4		r-4	SGN	LDY Z,X	180
181	B5		r-5	INT	LDA Z,X	181
182	B6		r-6	ABS	LDX Z,Y	182
183	B7		r-7	USR		183
184	B8		r-8	FRE	CLV	184
185	B9		r-9	POS	LDA Y	185
186	BA		r-:	SQR	TSX	186
187	BB		r-;	RND		187
188	BC		r-	LOG	LDY X	188
189	BD		r-=	EXP	LDA X	189
190	BE		r-	COS	LDX Y	190
191	BF		r-?	SIN		191
192	C0			TAN	CPY #	192
193	C1	,a		ATN	CMP(I),X	193
194	C2	,b		PEEK		194
195	C3	,c		LEN		195
196	C4	,d		STR$	CPY Z	196
197	C5	,e		VAL	CMP Z	197
198	C6	,f		ASC	DEC Z	198
199	C7	,g		CHR$		199

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
200	C8	,h	Reverse of ASCII	LEFT$	INY	200
201	C9	,i		RIGHT$	CMP #	201
202	CA	,j		MID$	DEX	202
203	CB	,k				203
204	CC	,l			CYP	204
205	CD	,m			CMP	205
206	CE	,n			DEC	206
207	CF	,o				207
208	DO	,p			BNE	208
209	D1	,q			CMP(I),Y	209
210	D2	,r				210
211	D3	,s				211
212	D4	,t				212
213	D5	,u			CMP Z,X	213
214	D6	,v			DEC Z,X	214
215	D7	,w				215
216	D8	,x			CLD	216
217	D9	,y			CMP Y	217
218	DA	,z				218
219	DB					219
220	DC					220
221	DD				CMP X	221
222	DE				DEC X	222
223	DF					223
224	EO				CPX #	224
225	E1				SBC(I),X	225
226	E2					226
227	E3					227
228	E4				CPX Z	228
229	E5				SBC Z	229
230	E6				INC Z	230
231	E7					231
232	E8				INX	232
233	E9				SBC #	233
234	EA				NOP	234
235	EB					235
236	EC				CPX	236
237	ED				SBC	237
238	EE				INC	238
239	EF					239
240	FO				BEQ	240
241	F1				SBC(I),Y	241
242	F2					242
243	F3					243
244	F4					244
245	F5				SBC Z,X	245
246	F6				INC Z,X	246
247	F7					247
248	F8				SED	248
249	F9				SBC Y	249

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
250	FA					250
251	FB					251
252	FC					252
253	FD				SBC X	253
254	FE				INC X	254
255	FF			π		255

Hexadecimal Conversion Table

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

5		4		3		2		1		0	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0		0		0		0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

6502 Instruction Set – Hex and Timing

		IMPLIED			ACCUM.			ABSOLUTE			ZERO PAGE			IMMEDIATE			ABS. X		
MNEMONIC		OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#
A D C	(I)							6D	4	3	65	3	2	69	2	2	7D	4	3
A N D	(I)							2D	4	3	25	3	2	29	2	2	3D	4	3
A S L					OA	2	I	OE	6	3	06	5	2				IE	7	3
B C C	(2)																		
B C S	(2)																		
B E Q	(2)																		
B I T								2C	4	3	24	3	2						
B M I	(2)																		
B N E	(2)																		
B P L	(2)																		
B R K		00	7	I															
B V C	(2)																		
B V S	(2)																		
C L C		18	2	I															
C L D		D8	2	I															
C L I		58	2	I															
C L V		B8	2	I															
C M P								CD	4	3	C5	3	2	C9	2	2	DD	4	3
C P X								EC	4	3	E4	3	2	E0	2	2			
C P Y								CC	4	3	C4	3	2	C0	2	2			
D E C								CE	6	3	C6	5	2				DE	7	3
D E X		CA	2	I															
D E Y		88	2	I															
E O R	(I)							4D	4	3	45	3	2	49	2	2	5D	4	3
I N C								EE	6	3	E6	5	2				FE	7	3
I N X		E8	2	I															
I N Y		C8	2	I															
J M P								4C	3	3									
J S R								20	6	3									
L D A	(I)							AD	4	3	A5	3	2	A9	2	2	BD	4	3
L D X	(I)							AE	4	3	A6	3	2	A2	2	2			
L D Y	(I)							AC	4	3	A4	3	2	A0	2	2	BC	4	3
L S R					4A	2	I	4E	6	3	46	5	2				5E	7	3
N O P		EA	2	I															
O R A								0D	4	3	05	3	2	09	2	2	ID	4	3
P H A		48	3	2															
P H P		08	3	I															
P L A		68	4	I															
P L P		28	4	I															
R O L					2A	2	I	2E	6	3	26	5	2				3E	7	3
R O R					6A	2	I	6E	6	3	66	5	2				7E	7	3
R T I		40	6	I															
R T S		60	6	I															
S B C	(I)							ED	4	3	E5	3	2	E9	2	2	FD	4	3
S E C		38	2	I															
S E D		F8	2	I															
S E I		78	2	I															
S T A								8D	4	3	85	2					9D	5	3
S T X								8E	4	3	86	2							
S T Y								8C	4	3	84	2							
T A X		AA	2	I															
T A Y		A8	2	I															
T S X		BA	2	I															
T X A		8A	2	I															
T X S		9A	2	I															
T Y A		98	2	I															

ABS. Y			(IND. X)			(IND)Y			Z.PAGE,X			RELATIVE			INDIRECT			Z.PAGE,Y			PROCESSOR STATUS CODES
OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	N V BDIZC
79	4	3	6I	6	2	7I	5	2	75	4	2										• • ••
39	4	3	2I	6	2	3I	5	2	35	4	2										• •
									I6	6	2										• • ••
												90	2	2							
												BO	2	2							
												FO	2	2							•
												30	2	2							
												DO	2	2							
												IO	2	2							
												50	2	2							
												70	2	2							
D9	4	3	CI	6	2	DI	5	2	D5	4	2										• ••
																					• ••
																					• ••
									D6	6	2										• •
																					• •
																					• •
59	4	3	4I	6	2	5I	5	2	55	4	2										• •
									F6	6	2										• •
																					• •
															6C	5	3				• •
																					• •
B9	4	3	AI	6	2	BI	5	2	B5	4	2										• •
BE	4	3																B6	4	2	• •
									B4	4	2										• •
									56	6	2										• ••
I9	4	3	OI	6	2	II	5	2	I5	4	2										• •
																					• •
																					• • • • • • •
									36	6	2										• ••
																					• ••
									76	6	2										• ••
																					• • • • • •
F9	4	3	EI	6	2	FI	5	2	F5	4	2										• • ••
99	5	3	8I	6	2	9I	6	2	95	4	2										
									94	4	2							96	4	2	• •
																					• •
																					• •
																					• •

INDEX

Analog meter - 84, 129
ASCII code - 4, 7
Assembler format - 37
Borders - 29, 54
Border around screen block - 54
'Carrace' - 32
'Cell' - 15
Character block display - 166
Character generator - 4, 7
Circle drawing - 123
'Clock' - 125
Cursor control characters - 14
Cursor control routines - 16, 18, 19
Display design - 9
Displaying character blocks - 163
Double density - 109
Double density line plot - 117, 122
Double density point plot - 112
Example 1 - 57
Example 2 - 66
Example 3 - 84
Example 4 - 85
Example 5 - 86
Example 6 - 87
Example 7 - 88
Example 8 - 103
Example 9 - 104
Example 10 - 111
Example 11 - 122
Example 12 - 123
Example 13 - 123
Example 14 - 124
Example 15 - 125
Example 16 - 129
Example 17 - 156
Example 18 - 159
Example 19 - 169
Example 20 - 171
Example 21 - 195
Example 22 - 195
Example 23 - 196
Example 24 - 201
Fill screen block - 50
Fine resolution horizontal bar plot, left to right - 68
Fine resolution horizontal bar plot, right to left - 70
Fine resolution horizontal line plot, left to right - 76
Fine resolution horizontal line plot, right to left - 78
Fine resolution vertical bar plot, bottom to top - 72
Fine resolution vertical bar plot, top to bottom - 74
Fine resolution vertical line plot, bottom to top - 80
Fine resolution vertical line plot, top to bottom - 82
Formatting - 17
Games programs - 19, 22, 30, 32, 163
'GRAPHICS' - 34
Graph plot - 85, 86, 88, 124
'Hangman' - 22

Horizontal bar - 46, 68, 70, 76, 78
Horizontal sync - 1, 4, 177
Inverted field characters - 29, 52
Kinetic art - 15
Large screen address - 142
Large screen clear - 146
Large screen display - 148
Large screen initialisation - 144
Large screen page display - 152
Large screen page save - 154
Light pen circuits - 180, 181, 183, 184, 185
Light pen construction - 182
Light pen machine code package - 187, 191
Light pen principles - 177
Light pen programming - 182
Light pen set up - 188
Listing 1 - 16
Listing 2 - 17
Listing 3 - 18
Listing 4 - 19
Listing 5 - 19
Listing 6 - 22
Listing 7 - 29
Listing 8 - 32
Loading machine code - 34
Machine code display - 9, 34
Machine code graphics package - 34, 46 to 168
'Macro characters' - 163
'Macro character' design - 164
Memory mapped - 3, 177
Meters - 84, 129
'Minefield' - 19
Moving character blocks - 163
Moving characters - 30
'Moving signboard' - 169
Multiple screen page displays - 139
POKE command - 8, 26 to 33
PRINT command - 8, 13 to 25
Raster scan - 1
Reverse field - 29, 52
Reverse field screen block - 52
Screen address calculation - 36, 40
Screen format - 11, 57
Scrolling - 93
Scroll down - 100
Scroll left - 96
Scroll right - 94
Scroll up - 98
Status line - 26, 57
String manipulation - 15
TAB - 13
Using machine code - 35
Variables - 39, 42
Vertical bar - 48, 72, 74, 80, 82
Vertical sync - 1, 4, 177
Video RAM - 4, 6
Word processor (simple) - 156

Notes

OFFICIALLY APPROVED BY COMMODORE

OFFICIALLY APPROVED BY COMMODORE